# NoSQL - A Silver Bullet?

**Jonathan Law**

**Abstract**

**Fred Brooks Jr. outlined four inherent problems in software engineering in his paper No Silver Bullet. Being written almost thirty years ago new technologies have been developed which have not been evaluated against these properties. This paper will be evaluating the features of NoSQL databases and look to see how they affect the problems, whether positively or negatively.**

**Keywords: Changeability, Complexity, Conformity, Invisibility, No Silver Bullet, NoSQL**

## I Introduction

IN THE paper No Silver Bullet, Brooks Jr. (1986, p.1) states that *"There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity"*. He went on to assert that the four intractable problems of software engineering are:

- Complexity
- Conformity
- Changeability
- Invisibility

His paper was published in 1986, almost thirty years ago. Since then new technologies and methodologies have been conceived. In this paper, NoSQL will be discussed, looking at the four inherent problems outlined by Frederick P. Brooks Jr. and if NoSQL has made any improvement on the problems. In addition, we shall compare NoSQL against RDBMS databases, as they saw prominence prior to the popularity of NoSQL. NoSQL differ from RDBMS as NoSQL is a non-relational database which does not support ACID transactions as opposed to RDBMS being relational and supporting ACID transactions.

NoSQL can be considered a trend as opposed to a technology but it is a trend which inspired the development of different database technologies. Databases which are considered to be part of the NoSQL family include key-value, document, graph and wide-column stores.

## II Complexity

*"Digital computers are themselves more complex than most things people build."* (Brooks Jr., 1986, p.3). Brooks Jr. identified complexity to be an essential property of software, and that to abstract away the complexity of software is to lose the essence of the software itself.

NoSQL databases are based on simple tables which host all data that have no correlation to each other. To manipulate and retrieve data, database administrators are given a primitive set of commands. Complex business functionality requires significant amount of design and programming to implement, although simpler business models may be able to suffice with the primitive command set supplied (Lombardo, Di Nitto and Ardagna, 2012, p.443).

Reduction of complexity in software projects may be aided or hindered by use of NoSQL depending on the functionality required from the database itself. Should the majority of functions required from the database be satisfied by the instruction set provided by NoSQL, then complexity of the software should be reduced as the schema is simpler compared to that of a RDBMS. However, complex queries are expcted to be executed often, then complexity may increase within software, as it requires extra work and programming to be implemented. Complex query programming can be difficult (Leavitt, 2010, p.13).

NoSQL offers features such as auto-sharding, replication and integrated caching natively. While this can be achieved with RDBMS, it is only capable of doing so through external software or development of systems to do so (MongoDB, 2013). This is an area where NoSQL is capable of reducing complexity. It seems that there is a trade-off about *where* the complexity lays within the system, rather than *how much* complexity there is. It is important to consider what is functionality is required from the database when choosing a database system, in order to reduce the complexity which the developers will face.

## III Conformity

*"Much of the complexity he must master is arbitrary complexity, forced without rhyme or reason by the many human institutions and systems to which his interfaces must conform."* (Brooks Jr., 1986, p.3).

Software systems must conform to many different standards and formats; There are laws and standards which specify multitudes of factors such as the storage of data (Data Protection Act 1998), while there are other systems and environments in place which the software may require to interface to, such as externalised APIs.

As databases are designed with the intent of storing data, it is important for such technologies to conform to the Data Protection Act (1998), which specifies a multitude of factors to ensure data security. It has been evaluated that NoSQL databases have common problems in allowing the encryption of data files as well as weak authentication between client and servers (Okman, et al., 2011), which concerns data security.

The arbitrary complexity that is enforced *"differ from interface to interface, and from time to time"* (Brooks Jr., 1986, p.4). As discussed in the Changeability section of this paper, NoSQL's enhanced ability to adapt to changes means that it is more suitable to conforming to sets of changing standards or requirements to interfaces when compared to that of a RDBMS. Its speed in executing these changes means there will be less downtime and as a result, less negative affect on the product. Against a static set of standards or requirements to interfaces which do not look to change often or regularly, such as that of laws or acts, NoSQL's changeability serves no advantages.

It may seem standards set by external organisations only add complexity to the system. One could assume that to do away with the standards would be beneficial. This would reduce the problem of conformity, and may have several benefits such as reduced complexity and speed of development. However, standards were primarily introduced for a reason. They have benefits such as encouraging process control, positive influence of the promotion of process repeatability and stablisation of activities (Tripp, L.L. and Voldner, P. 1995, p.108).

It would seem foolish to do away with standards when they were introduced to mitigate other problems. Perhaps the correct way to deal with conformity is to evaluate the management of standards and interfaces itself, instead of looking to a technology to mitigate against introduced overhead issues.

## IV Changeability

*"The software entity is constantly subject to pressures for change."* (Brooks Jr., 1986, p.4). Software is expected to change. Failed software is either scrapped entirely, or changed to suit the original requirements and functionality expected. Successful software outlives the machine it is written for, and is often adapted to its new host. Users attempt to use the software outside the confines of its original specified domain, and requires changes to increase suitability. Changeability is an expected behaviour of software.

NoSQL databases feature dynamic schemas which allow insertion of data without having predefined a schema for the database to adhere to. Key value stores are especially so in this respect as they are wholly schemaless. New values can be added at runtime without affecting any other data stored or the uptime of the system (Hecht and Jablonski, 2011). This complements agile methodologies in software projects well. It allows focus on the design and adapting to changes in the software, rather than the schema which supports the data. Changes in NoSQL schema have minimal friction (Sadalage and Fowler, 2012).

Comparing to that of RDBMS which use schemas which are defined before any operation can be performed. Making any changes which require a change in the schema, requires the schemas to be modified first and for the database to be migrated to the new schema. This may require a migration project in order to complete schema changes. Database schema migration can involve writing change scripts which need to be written from scratch for each change. This can be error-prone with slow turn about times, while also requiring that the database is unusable while the migration is taking place. This can be mitigated by using a copy of the database to host the service, while another version is being updated (Sadalage and Fowler, 2012). This fits in poorly with the ideals of agile methodology, including the aim to meet rapidly changing requirements (Livermore, 2007).

NoSQL's dynamic schema supports frequency and speed of change in a positive way. Changes in databases can be done quickly, scaling well with the size of the change to be made while having the capability to avoid down-times with the service and without the need of any scripts or migration projects. The changes can be made simply and effectively. This helps to reduce the effect of problems encountered due to the required and expected changeability in software.

## V Invisibility

*"Software is invisible and unvisualizable. The reality of software is not inherently embedded in space."*

(Brooks Jr., 1986, p.4). Software is difficult to visualise and as a result can be difficult to document software design. This becomes more difficult as software systems tend to integrate difficult technologies and have a need to convey and abstract different information to different people.

RDBMS make use of Entity Relationship Diagrams (ERD) to diagrammatically represent its schema. While the database is in operation, this schema will always be correct and consistent throughout the database. Data that does not conform to the database's schema will be rejected. ERDs are capable of abstracting away unnecessary information, such as data type, data length, constraints, etc. ERDs were designed to be used to represent different views of data from a single perspective (Chen, 1976).

Visualisation of a NoSQL database's model is made difficult by the differences in its data model (within the same database) as well as a lack of explicit schemas and querying language (Näsholm, 2012). While there may be an implicit schema throughout the database, data entered or modified does not necessarily conform to the implicit schema (Fowler, 2013). For example, an 'Age' field may be an integer for one user, but could be specified as a string of text for another. While this is beneficial in terms of changeability and flexibility within the system, it brings about complications when attempting to construct a visual representation of the database. This means there can be no single representation of the database that will be consistent throughout, even while in operation (Fowler, 2013).

UML Class Diagrams have been proposed and used as a way of constructing representations of NoSQL schemas. Delfosse (2012) has suggested how to map a graph store database to UML Class Diagrams, while some users of Stack Overflow (2013) have been seen using Class Diagrams for the purpose of representation of NoSQL databases. However this still does not tackle the problem of possible inconsistency throughout the database.

It can be said that RDBMS is superior in this aspect. The strict and rigid schema provides a useful way of representing the data storage model. Opposing this to NoSQL which is flexible and inconsistent, making it difficult to meaningfully convey the model in a diagrammatic format.

## VI Conclusion and Future Work

It seems that while the four inherent problems of software development are truly inherent, or at the very least not wholly solved with the introduction of NoSQL databases, it is important to note that the effects of these problems can be rebalanced to other areas.

While the dynamic nature of NoSQL promotes changeability, it does so at the cost of its visibility. Complexity has a balance shift, with complexity moved towards the manipulation and representation of the data, comparing to the complexity of RDBMS and its schema and relationship between data. Conformity also sees a similar shift. While NoSQL does not natively supporting strong encryption and security features, it does adapt to required changes well.

When deciding which database technology to use, one should consider the requirements of the system, and the advantages and disadvantages of the technology with the objective of reducing the effect of the inherent problems outlined. It may be possible to avoid the complexity introduced by NoSQL's limited instruction set if their model never requires the execution of complex queries. In the same manner, it may be more suitable to use RDBMS if the system is expected to see little change during its lifetime.

While it is difficult to say with complete certainty if any technology or methodology will wholly mitigate the four inherent problems of software development, NoSQL as a technology by itself, does not serve to be a "silver bullet" which Brooks Jr. proposed and does not appear capable of becoming one in the near future, due to its schemaless nature.

In the future, perhaps there will be a new emergent technology to drive databases to further become a silver bullet. Perhaps a redesign of databases will be required, meeting both the advantages provided by NoSQL and RDBMS databases.

## VII Acknowledgments

J.L Author would like to express gratitude to Dr Simon Scola for having led lectures as well as organising and participating in a review session of the quality assurance process alongside Ing Alex Fedorec. This gratitude also extends to other students who were present and gave feedback during the review session.

Thanks is also extended to those who agreed and participated in peer reviewing and proof reading this paper.

## VIII References

Brooks Jr., F.P., 1986, *No Silver Bullet - Essence and Accident in Software Engineering*

[pdf] University of North Carolina at Chapel Hill. Available at: <http://faculty.salisbury.edu/~xswang/Research/Papers/SERelated/no-silver-bullet.pdf> [Accessed 29 October 2013].

Lombardo S., Di Nitto E. and Ardagna D., 2012, *Issues in Handling Complex Data Structures with NoSQL databases* [e-journal] DEI Politecnico di Milano.Available through: IEEE Xplore Digital Library at: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6481064> [Accessed 29 October 2013]

Leavitt N., 2010. Will NoSQL Databases Live Up To Their Promises? *Computer*, [e-journal] 43(2). Available through: IEEE Xplore Digital Library at: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5410700>[Accessed 29 October 2013]

MongoDB, 2013. *NoSQL Databases Explained.* [online] Available at: <http://www.mongodb.com/learn/nosql> [Accessed 05 November 2013]

Tripp, L.L. and Voldner, P., 1995. *A market-driven architecture for software engineering standards* [pdf] Seattle, WA. Available at: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=525956> [Accessed 07 November 2013]

Hecht, R. and Jablonski, S., 2011. *NoSQL Evaluation - A Use Case Oriented Survey* [e-journal] University of Bayreuth, Germany. Available through: IEEE Xplore Digital Library at: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6138544> [Accessed 07 November 2013]

*Data Protection Act 1998.* (c.2). London: HMSO.

Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E. and Abramov, J., 2011. *Security Issues in NoSQL Databases* [e-journal] Available through: IEEE Xplore Digital Library at: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6120863> [Accessed 12 November 2013]

Sadalage, P. J. and Fowler M., 2010. *NoSQL Distilled A Brief Guide to the Emerging World of Polyglot Persistence.* Indiana: Pearson Education, Inc.

Livermore, J. A., 2007. *Factors that Impact Implementing an Agile Software Development Methodology.* [pdf] Walsh College. Available through: IEEE Xplore Digital Library at: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6138544> [Accessed 07 November 2013]

Chen, P., 1976. The Entity-Relationship Model - Towards a Unified View of Data. *ACM Transactions on Database Systems,* 1(1), pp.9-36.

Näsholm, P., 2012. *Extracting Data from NoSQL Databases* [pdf] Göteburg, Sweden. Available at: <http://publications.lib.chalmers.se/records/fulltext/155048.pdf> [Accessed 12 November 2013]

Fowler, M., 2013. *Schemaless Data Structures.* [online] Available at: <http://martinfowler.com/articles/schemaless/> [Accessed 08 November 2013]

Delfosse, V., 2012. *UML as a schema candidate for Graph Databases* [pdf] Available at: <http://ebookbrowsee.net/vincent-delfosse-uml-as-a-schema-candidate-for-graph-databases-pdf-d390375325> [Accessed 05 November 2013]

Stack Overflow, 2013. *MongoDB: How to represent a schema diagram in a thesis?* [online] Available at: <http://stackoverflow.com/questions/11323841/mongodb-how-to-represent-a-schema-diagram-in-a-thesis> [Accessed 05 November 2013]

## IX Glossary of Terms

**RDBMS:** Relational Database Management System. A type of database which typically uses SQL as a query language, where tables are created in a pre-defined schema and tables have cardinality with one another.
**Schema:** In terms of databases, can be compared to that of a plan or blueprint that the data stored in the database must follow. For example, 'Age' field must be an integer. Typically spoken about in reference to an RDBMS, with the exception of a dynamic schema which belongs to NoSQL.
**SQL:** Structured Query Language. A language designed for the purpose of performing create, read, update and delete operations within a database. Also allows for complex manipulation of data.
**API:** Application Programming Interface. Can be considered an interface which specifies how software

components should interact with each other.

**ERD:** Entity Relationship Diagram. A diagram used in RDBMS databases to depict tables (referred to as entities), the data they store, and the cardinality between tables.

**UML:** Unified Modelling Language. A standardized set of diagrams used to in software development capturing different views of software systems. For example a UML Class Diagram depicts the conceptual objects used in a system.