TECHNICAL OVERVIEW:
THE ARRIA NLG ENGINE

An inside look at how the NLG Engine combines cutting-edge techniques in data analytics and computational linguistics.

WWW.ARRIA.COM

The Science of Information Delivery
The Arria NLG Engine

Arria’s NLG Engine is a sophisticated software tool that combines cutting-edge techniques in data analytics and computational linguistics. These capabilities allow the Engine to convert large and diverse datasets into meaningful natural language narratives, helping solve the problems of dealing with Big Data. This paper looks under the hood of the NLG Engine to explain how the technology works.

**Analysis and Interpretation**

**Information Delivery**

DATA can be ingested from a wide variety of data sources, both structured and unstructured.

DATA ANALYSIS processes the data to extract the key facts that it contains.

DATA INTERPRETATION makes sense of the data, particularly from the point of view of what information can be communicated.

DOCUMENT PLANNING takes the messages derived from the data and works out how to best structure the information they contain into a narrative.

MICROPLANNING works out how to package the information into sentences to maximise fluency and coherence.

SURFACE REALISATION ensures that the meanings expressed in the sentences are conveyed using correct grammar, word choice, morphology and punctuation.

NARRATIVE can be output in a variety of formats (HTML, PDF, Word, etc.), combined with graphics as appropriate, or delivered as speech.
Introduction

Natural Language Generation is all about taking data and turning it into written or spoken language. This is indisputably the best way to provide actionable analytics. It’s how we as a species have communicated information and calls-to-action for tens of thousands of years. It’s a finely honed device that evolves as we evolve, and it marks us out from other species. It’s the best means by which we answer the requests ‘Tell me what I need to know’ and ‘Tell me what I should do’.

The essence of this amazing capability has now been distilled down into software. Based on decades of research into language and linguistics, our NLG Engine is able to take data, analyse it for the insights it contains, and present those insights in language that you would believe had been written by a human.

This isn’t mail merge or document assembly. It isn’t filling in slots in templates. It’s about taking the data source and using knowledge of the domain to massage and aggregate that data; identifying packets of information that can be expressed linguistically; then using rich knowledge of language to work out how best to express that information in text or voice.

Our patent-pending technology embodies the knowledge of experts in the domains in which we operate, and combines this with our own knowledge of how language and communication works, to produce articulate, fluent and coherent text. We call this process multilevel NLG, to distinguish it from the simpler approaches to content creation used by others.

The Arria NLG Engine provides a powerful and flexible software architecture, one that has been refined through the many decades of collective experience that our scientists and engineers have acquired in building rich NLG applications in a variety of domains.
# Table of Contents

The Problem and a Solution ........................................................................................................ 5
How the Arria NLG Engine Works .......................................................................................... 6
The Arria NLG Software Architecture .................................................................................... 7

Analysis and Interpretation

Data Analytics ........................................................................................................................... 9
Data Interpretation .................................................................................................................. 10

Information Delivery

Document Planning .................................................................................................................... 11
Microplanning .......................................................................................................................... 13
Linguistic Realisation ............................................................................................................... 14

Arria NLG Configuration and Deployment .............................................................................. 15
An Arria NLG Engine Use Case ............................................................................................... 19
The Benefits of Using Arria NLG .............................................................................................. 20
Arria’s NLG Scientists ............................................................................................................. 21
The Problem

Today, data is everywhere. Every computer system and every piece of technology is capable of producing vast amounts of data, from the humblest personal fitness device that counts your steps to the massive engines on modern passenger airliners that send constant streams of performance data from multiple sensors back to base for health monitoring. And major corporations in every information-based industry, from finance through telecom providers to search engine giants, rely on endless streams of transactional data to serve as their lifeblood.

All of this data is a good thing. It opens up massive potential for diagnosis, performance improvement, or any of a wide range of other purposes that have a direct impact on the bottom line. So the data can help us solve problems. But here’s the paradox: the data itself has become a problem because there is so much of it. As is now so often acknowledged in the technical press, we are drowning in data, buried so deep that we can’t make sense of the data that is being fed to us.

“The real problem is the shortage of the expertise required to make sense of Big Data.”

In fact, the problem is not really the data itself. The real problem is a shortage of the expertise required to make sense of the data and to communicate the meaning of that data to those who need to act. This problem is widespread: numerous studies across a variety of industries have bemoaned the acute shortfall in human resources that prevents us carrying out analysis of data and delivering the information that it contains.

A Solution

To make use of the raw data being provided to us by these myriad sources, there are two things we have to do. First, we have to analyse the data to extract useful information and insight from it. Second, we have to communicate that information and insight to those who need to act upon it, in terms that they can understand. Put simply, we have to tell the data’s story.

This is what a human analyst does, and—up until now—it has only been human analysts who have been able to do this. But now the same skills can be embodied in software. This is what the Arria NLG Engine does.
How the Arria NLG Engine Works

THE ARRIA NLG ENGINE has two major components, corresponding to the two skill sets possessed by any expert who is both an analyst and a communicator. First, the Engine contains a Data Analytics and Interpretation stage, whose purpose is to take the various sources of data that need to be explained, and to extract and deduce from this data important facts and insights that should be communicated to an interested party. The results of this process are informational units we call messages.

“Arria has a solution that is capable of automatically generating truly articulate text from Big Data.”

The second stage of processing in the NLG Engine takes these messages and works out how to communicate the information they contain in an articulate and coherent manner, using natural language text and, where appropriate, graphical representations of the data. Voice output can also be produced. Each of these two major processing stages involves a number of substeps, described below. We call this approach multilevel NLG, since it makes use of multiple distinct levels of representation of the information to be conveyed. This is to be contrasted with simpler template-based technologies that simply suck data elements from a source and drop them into slots in a predefined text. In the Arria NLG Engine, the construction of a textual narrative is a careful, deliberate process at many levels. Each level provides for flexibility and variation in how the end result will look, so that a single data source can give rise to many different explanatory texts, tailored for the needs and interests of the specific audience.

This sophisticated multilevel NLG approach is the result of academic research that Arria’s lead scientists have carried out over the last 30 years, positioning them as authorities in the field. Through extensive experience in building NLG applications, our team has learned where simpler techniques fail, and has crafted a solution that is capable of generating truly articulate text.
NLG Software Architecture

The purpose of an NLG system is to take information expressed in some non-linguistic format and to express that information in a natural language. The input data may be:

- contained in a spreadsheet or database
- presented in tabulated log messages or some other formally well-defined structures
- encoded in a ‘knowledge representation’ such as the RDF triples that make up the semantic web.

The data may be primarily numerical, or may include other symbolic content.

The output of the NLG system is typically text, but may also be in the form of speech; and the textual output may be part of a more complex presentation that also includes graphical content (in which case we have what we call multimodal generation).

“Our multilevel NLG technology is many times more powerful than simpler approaches.”

This transformation requires applying a number of processes to the input data, such as:

- ordering and structuring the information;
- selecting some subset of the information for presentation;
- deriving new information on the basis of the input data; and
- identifying useful abstractions that provide a higher-level description of the data.

The information is then mapped into natural language output, typically via a sequence of operations of ever-finer linguistic granularity, from overall text organisation through to choosing the correct inflections for words.

The NLG Engine is organised as a pipeline of components that progressively refine the input information until it can be expressed in language; other configurations are possible for specialised use cases.

**FIGURE 1** shows the pipeline of components and summarises what each does. This document spells out in more detail the various types of processing these components carry out.
Multilevel NLG Software Architecture

DATA can be ingested from a wide variety of data sources, both structured and unstructured.

DATA ANALYSIS processes the data to extract the key facts.

DATA INTERPRETATION makes sense of the data, particularly from the point of view of what information can be communicated.

DOCUMENT PLANNING takes the messages derived from the data and works out how to best structure the information they contain into a narrative.

MICROPLANNING works out how to package the information into sentences to maximise fluency and coherence.

LINGUISTIC REALISATION ensures that the meanings expressed in the sentences are conveyed using correct grammar, word choice, morphology and punctuation.

NARRATIVE can be output in a variety of formats (HTML, PDF, Word, etc.), combined with graphics as appropriate, or delivered as speech.
Analysis and Interpretation

Our Data Analytics and Interpretation component makes use of the full armoury of techniques that you would get from a pure-play data analytics vendor—explicit rule-based reasoning, machine-learned data mining, pattern recognition, space and time series analysis, and criticality assessment—but with an extra twist: every part of our analytics is oriented towards deriving information that can be communicated using language. That causes us to detect patterns, trends and concepts that elude other approaches. We can also integrate and combine the output of other analytics providers where available.

Our data ingestion machinery is capable of consuming data in a wide variety of formats. There are typically two processes we undertake to make sense of this data, which we refer to as Data Analytics and Data Interpretation.

“The purpose of NLG is to add value by making the content of the raw data more accessible to a human user.”

DATA ANALYSIS

The data that is provided as input to an NLG system is not usually created with linguistic presentation in mind; it is more likely intended for consumption by some other software process, or is intended to be used in constructing graphs and charts.

A common type of data consumed by NLG systems is time-series data, capturing the variation across time of entities such as stock prices, company profits, rainfall amounts, temperature, or employment statistics. Data may also be spatial, indicating variation across locations (for example, rainfall in different geographical areas), or spatio-temporal, combining variations in both dimensions (for example, rainfall across time in different geographical areas). We refer to any of these types of data as raw data.

In all these cases, the purpose of NLG is to add value by making the content of the raw data more accessible to the user. The first step in this process is one of data analysis, where we detect patterns and trends in the data that allow us to capture what is going on at a higher level of abstraction than the raw data itself.
Example: A time-series dataset may contain tens of thousands of individual records describing the temperature at various points on a component piece of machinery over the course of a day, sampled once every two or three seconds. Trend analysis might allow us to identify that the temperature changes in a characteristic way throughout certain parts of the day, providing insights into potential failure scenarios for the machine in question.

There are a wide range of existing techniques for analysing time-series data to identify such abstractions, which can be thought of as ways of compressing the detailed raw data into a much smaller number of higher-level observations. The techniques required depend on the domain and the properties of the data; for some domains it may be appropriate to develop new techniques that go beyond those offered by standard statistical tools.

“Messages are intended to be language independent, so that they might equally well be conveyed in English, French or some other natural language.”

DATA INTERPRETATION

The abstractions produced at the data analysis stage are based on what the raw input data makes available. At the other end of the process, the analysis of a corpus of human-authored texts, or, if no such texts are available, a requirements analysis process, will determine the kinds of things we want to communicate about the domain. We refer to these as the different types of messages we might want to convey. A message typically corresponds to a simple relationship between a small number of entities that could be expressed via a simple sentence (although it may ultimately be realised by some other means). Note that messages are intended to be language-independent, so that they might equally well be conveyed in English, French or some other natural language.

The data interpretation process then populates or instantiates messages on the basis of the abstractions identified during the data analysis stage.

The output of this stage consists of a set of instantiated messages that can be used to build a report. The set of messages derived from the data is referred to as the knowledge pool. The knowledge pool may also contain information that indicates how these messages are related to each other; for example, one message may be marked as describing a state of affairs that is assumed to cause the state of affairs described in some other message.
Information Delivery

OUR INFORMATION DELIVERY component uses both text and graphics to communicate the information derived by our analytics tools. It’s here that the deep linguistic knowledge our Engine uses is embodied in software, and it’s worth elaborating on the power and capabilities of each part of this Engine. Many of the functionalities here are recognisable as things that we do every time we write or speak, but we shouldn’t let that familiarity blind us to just how complex these processes really are.

We stratify the particular kinds of linguistic knowledge our NLG Engine uses into three distinct levels, making our multilevel NLG technology many times more flexible and powerful than simpler approaches.

“We stratify the particular kinds of linguistic knowledge our Engine uses into three distinct levels, making our multilevel NLG technology extremely flexible.”

DOCUMENT PLANNING

Our Document Planning module knows all about building narratives: what information has to be conveyed and in what order, with those decisions often being dependent on who the audience is. Just like us, the software has to think ahead to what it’s going to say so that it doesn’t talk itself into a corner: it considers what information needs to be conveyed, and what information can safely be omitted. It puts the most important information first. It understands conventions about text structure that are second nature to us—like not giving away the punch-line before the end of the joke.

The purpose of the document planning stage is to use the messages available in the knowledge pool to tell a story about the data. The genre of the reports to be generated will dictate what information should be conveyed and in what order it should be presented, or at least place constraints on acceptable orderings.

Conceptually, document planning involves two tasks. First, we need to select the subset of the messages in the knowledge pool that should be reported in the text to be generated. In some cases, all the messages in the knowledge pool may need to be reported, but this is not always appropriate. Once we have determined what content needs to be expressed in the document, we have to provide an overall organisation for that content in text; this is achieved by a document structuring process.
In practice, these two tasks are usually folded together into one process. The required information about content and ordering can be encoded in different ways, but one of the most common and transparent is to use a text schema. This is essentially like a grammar rule, describing how a number of elements should be arranged together; but instead of organising words as in a standard grammar rule, a schema organises messages.

Each text schema contains information that determines what messages should be used to instantiate that schema. At the same time, the messages available in the knowledge pool determine what text schemas can be instantiated. It can be the case that multiple alternative schemas can be used, in which case some mechanism for choosing amongst them is required; for example, one might choose the schema that consumes the greatest proportion of the knowledge pool.

“Our Information Delivery component uses both text and graphics to communicate the information derived by our analytics tools.”

Depending on the circumstances, we use document planning processes that are either top-down or bottom-up, or some mixture of the two. Schema-based approaches are top-down: we start out with an overall plan for what we want the text to look like, and see if we have the required messages in the knowledge pool to instantiate this plan. Our document planning can also operate in a more bottom-up fashion that is not unlike solving a jigsaw puzzle: we start out with some key message that needs to be conveyed, then we see if we can build up a coherent text by growing outwards and connecting further messages to it. Coherence here is generally defined in terms of formally-specified constraints on the combinations of messages: for example, a certain kind of message may be able to serve as an elaboration of some other message, or as a justification for the information in that message. These relationships between messages are often referred to as rhetorical relations.

The result is a tree-structured object that we refer to as a document plan. The leaves of the tree are messages, and the intermediate nodes indicate how the subordinate nodes are related to each other.
MICROPLANNING

Document planning provides us with an overall organisation for the text, supporting the ordering of presentation and the clustering of messages into paragraph-sized chunks. As mentioned above, messages are essentially language-independent; so, although a message may contain symbols that are reminiscent of English words, this is just a convenience for the software developer. They are best understood as concepts, and so the particular words to be used in expressing these concepts still have to be selected. Also, although each message could at this point be expressed as a simple sentence, typically this would result in a low-quality, unsophisticated text.

“Information is expressed in the natural language of choice in accordance with the grammar rules of that language.”

These concerns are addressed by the microplanning stage, which carries out three specific tasks:

1. **LEXICALISATION**
   This involves choosing the particular words to be used in expressing the concepts in the messages. This is done by looking up the concepts in a lexicon, which may specify constraints that allow contextually-based choice between different alternative lexicalisations. For example, the context of use may determine whether we refer to people as ‘clients’, ‘customers’, or ‘users’; or whether we refer to the weather as being ‘humid’ or ‘muggy’.

2. **AGGREGATION**
   This involves determining whether two or more messages can be combined together linguistically in a more complex sentence. Languages offer a range of ways of doing this, often constrained by the rhetorical relations that hold between the messages to be conveyed. So, for example, instead of saying ‘Apple stocks rose by 5.1%. Microsoft stocks rose by 5.3%.’ we might say ‘Apple and Microsoft stocks both rose by just over 5%.’ Aggregation thus makes use of a collection of strategies that match patterns found in the messages; since these messages are already nodes in a tree-structured document plan, aggregation generally involves structural modifications to that tree.

3. **REFERRING EXPRESSION GENERATION**
   This involves choosing how to refer to some entity so that it can be unambiguously identified by the reader. When referring to individuals or places, for example, this might be achieved by repeated use of a proper name, but such a strategy generally results in a text that lacks fluency. So, pronouns or shortened descriptions are typically used, but these must be chosen in such a way as to avoid possible misinterpretations of the text. Consider how the president of the US is referred to in the following text: ‘President Obama met Prime Minister Cameron late yesterday. The President re-iterated that he would...’ This requires taking into account what has been referred to already, using information stored in a discourse history.
LINGUISTIC REALISATION

As a result of the microplanning stage, we now have a document plan that provides various types of information about the text to be produced:

• We know the overall structuring and ordering of the material to be conveyed.
• We know what rhetorical relations hold between the constituent parts of the document plan.
• We know how the messages are combined together into sentences.
• We know what words and phrases are to be used to refer to entities and concepts in the messages.

The final step is for this information to be expressed in the natural language of choice in accordance with the grammar rules of that language: this is the realisation process. To carry out realisation, we process the discourse plan by expressing each leaf node in the document plan in natural language. Each leaf node typically corresponds to a sentence, but in some cases (for example, headings in reports) it may correspond to a noun phrase or some other linguistic structure. Each leaf node either contains a single message or, if an aggregation strategy has been applied, two or more messages that have been combined.

“The realisation process provides a way of mapping from the message structures into natural language structures.”

The realisation process makes use of a grammar which specifies the valid syntactic structures in the language, and provides a way of mapping from the message structures into natural language structures. The realisation process also carries out morphological processing, ensuring that the correct inflections are used on words (so that, for example, plurals are properly formed and verb tenses are used correctly); and it also carries out orthographic processing, whereby punctuation marks are inserted, the first words of sentences are capitalised, and so on.

Realisation may also insert tags into the text that control its presentation. Internally, we use a proprietary Rich Output Format to carry structural information, and this is then converted into the appropriate format for publication. For example, HTML tags may be incorporated to generate web pages, or SSML tags may be incorporated to control stress and intonation in the case of synthesised speech.
Arria NLG Configuration and Deployment

So far we’ve described how the NLG Engine works, taking in raw data, applying multiple levels of processing, and producing as output narrative texts that explain that data appropriately for the audience at hand.

From a software configuration and deployment perspective, each stage of processing within the NLG Engine—the Data Analysis, the Data Interpretation, the Document Planning, the Microplanning and Realisation—is itself a software module that is driven by a set of rules. The core modules are the same in every application of the Arria NLG technology, but the particular rules in use define the behaviour of the Engine for that application.

We think of this architecture as being something like a pyramid, with our Core NLG Engine at the base representing upwards of 60% of the code in any given application. The remainder of the application consists of the configuration rules. In any given use of the NLG Engine, those rules are of three kinds (Figure 2).

“The core modules are the same in every application of the Arria NLG technology, but the particular rules in use define the behaviour of the Engine.”

First, there are general purpose rules that we use in almost every application of the Engine. We call this collection of rules the Core Engine Ruleset. These capture knowledge about data processing and linguistic communication in general, independent of the particular domain of application of the Engine.
Second, there are rules which encode knowledge about the specific industry vertical or domain in which the Engine is being used. These rules, which we call Vertical Rules, are collected together into what we call a Vertical Pack. Our Vertical Packs are constantly being refined via ongoing development, embodying knowledge about data processing and linguistic communication which is common to different clients in the same vertical.

And finally, there are rules which are specific to the client for whom the Engine is being configured. We call this collection of rules the Client Ruleset. These rules embody the particular expertise in data processing and linguistic communication that are unique to the client’s application.

“The modularity of this architecture means that the configuration of a new application for an existing client can be very fast.”

The proportions of each kind of rule vary by component. So, for example, we tend to find that a large proportion of the rules used by the Data Analytics processor are Core Engine rules, reflecting the fact that many domains have similar low level data processing requirements. Similarly, the Realisation process uses grammatical and morphological knowledge that is common to many domains—sentences have the same kinds of structure regardless of what is being talked about, with some notable exceptions such as the telegraphic language of weather forecasts—so most of the rules used here are Core Engine rules.

On the other hand, the Data Interpretation stage tends to use a larger proportion of Client Rules, reflecting the fact that in many instances of use, the power of the Engine is magnified by incorporating the client’s own proprietary knowledge about how to make sense of the data. Similarly, the Document Planner makes use of Vertical Rules and Client Rules to capture the particular idiosyncrasies of how information is reported for each specific use case.

The modularity of this architecture means that the configuration of a new application for an existing client can be very fast, since much of the knowledge used by the Engine will remain the same.

**HOW WE CONFIGURE THE ENGINE**

As implied above, deploying the NLG Engine for a particular use case is essentially a configuration task, where we provide the specific rulesets that will be required to enable the Engine to produce the desired reports. To do this, we have to capture the knowledge needed to build the Client Rulesets for each of the component processors.
Knowledge acquisition is the process of extracting, structuring and organising knowledge, typically from human experts, so that it can be used in software. We do this using many of the techniques that are used in the context of other artificial intelligence software development processes, including task observation and both free-form and structured interviews. But unique to our process is our Corpus Analysis Methodology, where we use a structured approach to analysing existing reports produced by human authors in order to uncover the tacit knowledge involved in their creation. Many clients comment on the additional advantage this brings in formalising knowledge that was previously only implicitly available.

Being able to access an existing corpus of human-authored reports and their corresponding data is the ideal way to identify and capture the required knowledge, but we acknowledge that in many situations there will not be an existing corpus of such reports. In these circumstances, our business analysts work with the client to develop target texts that demonstrate the kinds of output that are desired; the automatic generation of these target texts then provides a benchmark for determining progress towards a fully configured system.

“In some situations, the client will prefer a waterfall development approach, where a more detailed design is produced at the start of the project.”

By using this combination of knowledge acquisition techniques, we are able to build up the Client Rulesets that will be required for each component processor. Our development methodology is based on the agile model, so that we can show the client results early and often, allowing corrective feedback that steers the configuration process towards the Engine producing exactly the kinds of texts that the client wants to see.

In some situations, the client will prefer a waterfall development approach, where a more detailed design is produced at the start of the project. We’re also comfortable working in this mode, although we do encourage periodic meetings with the client’s Subject Matter Experts as the work proceeds, to ensure that the configured application meets the requirements. It is our experience that requirements can change in subtle ways as the client becomes aware of the full power of what is possible using the NLG Engine.
INTEGRATION AND CHANGE MANAGEMENT

The NLG Engine does not operate in a vacuum. It needs to be integrated into your systems, and it needs to be appropriately positioned within your workflow. This means being aware of both people issues and technical issues.

User involvement is preferred from an early stage, to ensure that those who will actually use the technology can participate in defining how it will fit in, and can see the benefits the technology brings. The better the fit of the technology design for existing workflows, the more likely the software will be embraced. At the same time, introducing an NLG solution often allows client employees to point out ways in which existing processes can be improved.

Our deployment team is experienced in change management, enabling them to smooth the process of melding cutting-edge technology with the client’s work practices.

Our aim is to make the integration of the technology as seamless as possible. Interfaces—both for ingesting data from the client’s data sources and for publishing reports back into the client’s environment—are established early on. Ancillary services such as logging and monitoring are also identified and decided early so that there are no surprises around the impacts of integration.

“We offer two primary deployment solutions: The configured NLG Engine can be accessed as a cloud-based service, or it can be installed locally at the client’s site.”

DEPLOYMENT AND HOSTING

Once configured, our normal procedure is to work through an agreed User Acceptance Process to ensure that the application is appropriately and thoroughly tested before being deployed.

We offer two primary deployment solutions: The configured NLG Engine can be accessed as a cloud-based service, or it can be installed locally at the client’s site. Our cloud-based service elastically takes care of changes in demand and ensures high performance. Local hosting may be more appropriate where the client’s data is required to stay onsite.

Whichever hosting solution is chosen, our licensing terms are structured to ensure that the NLG Engine always provides maximum value. Updates of the Core Engine are released on an agreed schedule, and the rulesets used by the Engine are updated to take account of evolving best practice in your organisation: this keeps the Engine’s outputs fresh and relevant.

The knowledge and expertise of your most experienced experts are permanently and continuously captured on an ongoing basis, protecting these valuable assets and sharing them to mitigate organisational knowledge loss.
An Arria NLG Engine Use Case

The Arria NLG Engine has been successfully deployed by a major player in the oil and gas industry. In the extractive industries, it's all about uptime: your profits are bounded by what you can pump or dig out of the ground, and the last thing you want is for your machinery to go offline, resulting in lost productivity. But as we’ve seen on numerous occasions in recent years in this industry, you also can’t be too safe: if it looks like there might be a problem, you don’t want a disaster, so you switch things off.

“Arria NLG software is designed to be updated continuously to reflect current lessons and experience.”

In our client’s case, we have massive pieces of machinery that are loaded up with sensors attuned to the slightest variations in movement, temperature and performance. These sensors spit out significant quantities of data; and they feed into analytics machinery that triggers alerts when important combinations of thresholds are breached. Because of the need to play safe, alerts are triggered with large safety margins. However, it can take a highly experienced engineer several hours to dig into the background and history of the alert situation to determine whether the problem is real, or just a false alarm. The scarcity of the human resources inevitably means that some potentially false alarms can’t be explored. This means that there are occasions when equipment is switched off when it may not have to be, with a consequent loss of uptime.

ENTER THE ARRIA NLG SOFTWARE ENGINE

We have embodied in our technology the same analytical expertise that the highly experienced engineer uses in determining what is really going on when an alert is triggered. But instead of taking several hours to construct this situational awareness, our application does it in several minutes, often taking account of a depth of data that would be beyond the human expert’s capabilities. We then present the results of that analysis in a report that communicates the situation to the other engineers involved in dealing with the situation, and we go further still: based on the information in the report, we generate a recommendation for action, again embodying the expertise of the most highly-skilled engineers. All of this is communicated in natural language - fluid, concise and articulate. And we can do this 24 hours a day, 365 days a year, with each analysis and report taking a tiny fraction of the amount of time a human would take to write it. We can replicate the functionality endlessly.
The Benefits of Using Arria NLG

THE ARRIA NLG ENGINE is designed to be updated continuously to reflect current learnings and experience. It can be programmed to tailor and deliver its messages to different audiences simultaneously—e.g. to an operator, to an engineer and to a manager—and it can work in prodigious volumes.

“Instead of taking several hours to construct situational awareness, our NLG application does it in several minutes.”

KNOWLEDGE CAPTURE

The ‘best practice’ knowledge and expertise of your most experienced experts are permanently and continuously captured. With NLG, these valuable assets can be protected and shared to mitigate organisational knowledge loss.

REPORT AUTOMATION

Comprehensive reports are automatically produced, reading as if written by the most experienced analysts—enabling companies to scale and replicate their internal expertise.

DECISION SUPPORT

By capturing the background analysis for decision-making in the reports, NLG frees up experts’ time to focus on higher value issues, and guards against experts inadvertently missing or ignoring key information when they are under pressure or otherwise stressed. NLG reduces decision timeframes from hours to seconds.

KEY BENEFITS:

- Greater consistency in the logic, structure and content of diagnostic narratives
- Experts are freed from laborious, repetitive data analysis to focus instead on value-added work
- Subject matter experts’ performance is enhanced by instant NLG Engine support
- Faster analysis of situations and response times
- Faster writing and dispatch of reports
- Increased production
- Scaling of expert resources
Arria’s NLG Scientists

The founding scientists of Arria NLG are international leaders in NLG research. They have played a pivotal role in the development of natural language generation over the past thirty years. Collectively they represent a significant proportion of the world’s natural language generation expertise with the lion’s share of experience in commercialising natural language generation by applying it to solve real-life problems in an increasingly data-driven world.

“Arria NLG has amassed the greatest concentration of natural language generation expertise in a single organisation.”

Our scientists have organised conferences, set up NLG research centres, written hundreds of research papers, presented at scientific meetings, supervised PhD students and held professorial positions at prominent universities around the world.

In addition to their academic research, our NLG scientists have sought out complex real-life problems that NLG theory can solve. This means they not only developed the theory behind NLG software systems—they also built the systems for its deployment.

Today, with the scientific leadership of Professor Ehud Reiter, Dr. Yaji Sripada and Dr. Robert Dale, Arria NLG boasts the greatest concentration of natural language generation knowledge and expertise ever amassed in a single organisation.
Prof. Reiter is one of the world’s leading experts in natural language generation (NLG) research. He obtained a PhD in Computer Science from Harvard in 1990, and worked at the University of Edinburgh and CoGenTex before joining the University of Aberdeen in 1995. At Aberdeen he founded the Aberdeen NLG research group, which became one of the largest and most active NLG research groups in the world.

He has over 150 academic publications in natural language generation and medical informatics. His best known publication is *Building Natural Language Generation Systems*, a book he wrote with Dr Robert Dale (Arria’s Chief Technology Officer), which is widely used as a textbook for NLG. In 2009 he co-founded Data2text Ltd to commercialise the NLG technology developed at the University of Aberdeen; when Arria acquired Data2text in 2013, he became Arria’s Chief Scientist.

Dr Sripada is a leading natural language generation (NLG) expert and Senior Lecturer in Computing Science at the University of Aberdeen. He obtained a PhD at the Indian Institute of Technology in Chennai in 1998, and subsequently worked for 3 years in the software industry with Tata Consultancy Services (TCS) in India before moving to Aberdeen in 2000. Dr Sripada is widely published on the subject of natural language generation.

His research focuses in particular on the integration of data science and NLG, a novel hybrid technology space he calls articulate intelligence. In 2009 he co-founded Data2text Ltd to commercialise the NLG technology developed at the University of Aberdeen; when Arria acquired Data2text in 2013, he became Arria’s Chief Development Scientist.

Dr Dale is an internationally recognised leader in natural language generation (NLG) research and development. Since receiving his PhD from the University of Edinburgh in 1989, he has taught at universities in the UK and Australia. He has published over 160 papers and authored or edited seven books on a wide range of topics in natural language processing. His best known publication is *Building Natural Language Generation Systems*, a book he wrote with Professor Ehud Reiter (Arria’s Chief Scientist), which is widely used as a textbook for NLG.

From 2003 to 2012 he served as editor-in-chief of *Computational Linguistics*, the preeminent international journal on all aspects of natural language processing. In 2012 Dr Dale stepped down as Professor in Computational Linguistics at Sydney’s Macquarie University to join Arria, where he is now responsible for ensuring that the company’s technology offerings remain at the cutting edge.
The market leader in real-time data storytelling

REQUEST A DEMO

WWW.ARRIA.COM

Arria NLG plc is a company registered in England and Wales having its registered office at Space One, 1 Beadon Road, London W6 0EA, United Kingdom with registered number 07812686. Arria NLG Software Engine, Arria NLG Engine and NLG Engine are trademarks of Arria NLG plc. Company names and company logos are trademarks of their respective owners. Entire Contents © 2016 by Arria NLG plc with all rights reserved.