IDIOM

Taming the IT Beast

INCREASE BUSINESS AGILITY
REDUCE SYSTEM COMPLEXITY

IDIOM
PURE DECISIONING

Contents

Synopsis

Over the past 15 years IDIOM has conceived, evolved, and demonstrated the effectiveness of its 'decision centric' development approach, which leverages both *decisioning* and *agile* approaches to dramatically simplify and strengthen commercial systems development. This advertorial describes the IDIOM products and how they can be used to implement the decision centric approach.

Development of systems using the approach, and the IDIOM tools, has been repeatedly shown to:

- simplify systems and their related development practices to reduce development time, cost, and risk by substantial amounts
- increase business agility
- improve the alignment of business policy and computer systems
- improve runtime performance and reduce operational resource requirements.

The evolution of the approach has been chronicled in a series of published articles that remain available as published.
For convenience, we have included a précis of the more important articles in the section titled 'The IDIOM approach – a chronicle'. These précis' provide a shortcut into the background of the methodology and its rationale. Links to the original articles are also provided for the interested reader.
The IDIOM products are derived from, and closely aligned with, the concepts described in the articles, and together they provide a cost effective and risk averse path to successfully building decision centric systems. An outline of each of the IDIOM tools is offered, followed by discussion on how they are used to support the decision centric development approach – that is, the IDIOM approach.
This is followed by further discussion about the implications of *nimble, continuous, perpetual* development, including versioning and effective dating, and how this is addressed by the tools.
The approach can address significant *problem complexity*, huge *operational scale*, and a wide range of *business domains*.
Real projects are presented that demonstrate these capabilities.
Everything presented herein is based on common sense and a desire to simplify our overly complex IT world; there is no magic. But it does work, and it does deliver outstanding results.

Please contact us if you would like to discuss any aspect of this publication, to see a product demonstration, or to contemplate a proof of concept. We would be happy to oblige.

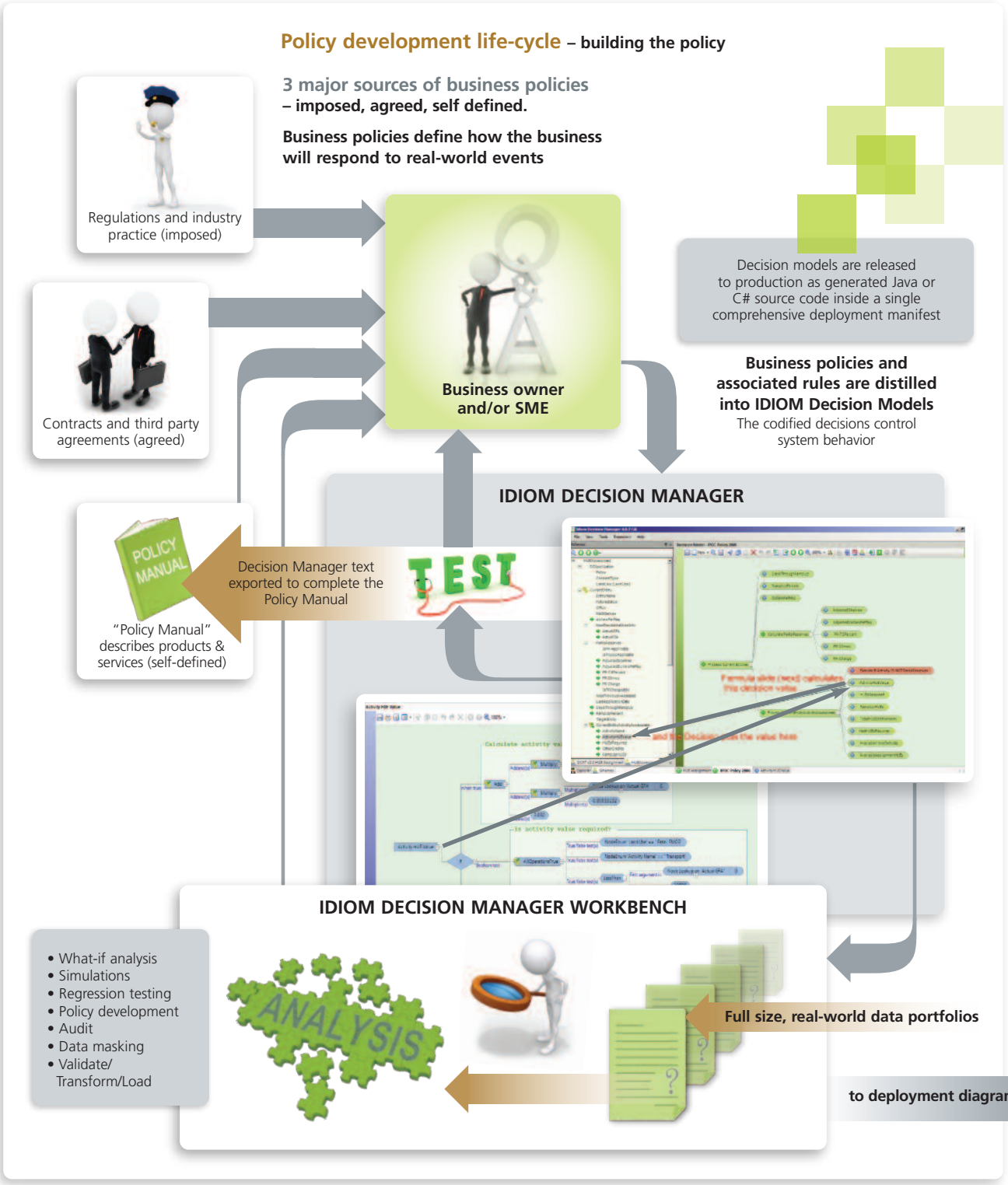
Mark Norton
CEO and Founder, IDIOM Ltd.

Author's Contact Details
Mark Norton, CEO and Founder, IDIOM Limited
Office +64 9 630 8950, Mob +64 21 434669,
After Hours +64 9 817 7165, Australia Free Call 1 800 049 004
2-2, 93 Dominion Road Mt Eden, Auckland 1024
PO Box 60101, Titirangi, Auckland 0642, New Zealand
email mark.norton@idiomsoftware.com
Skype Mark.Norton

The IDIOM approach in pictures

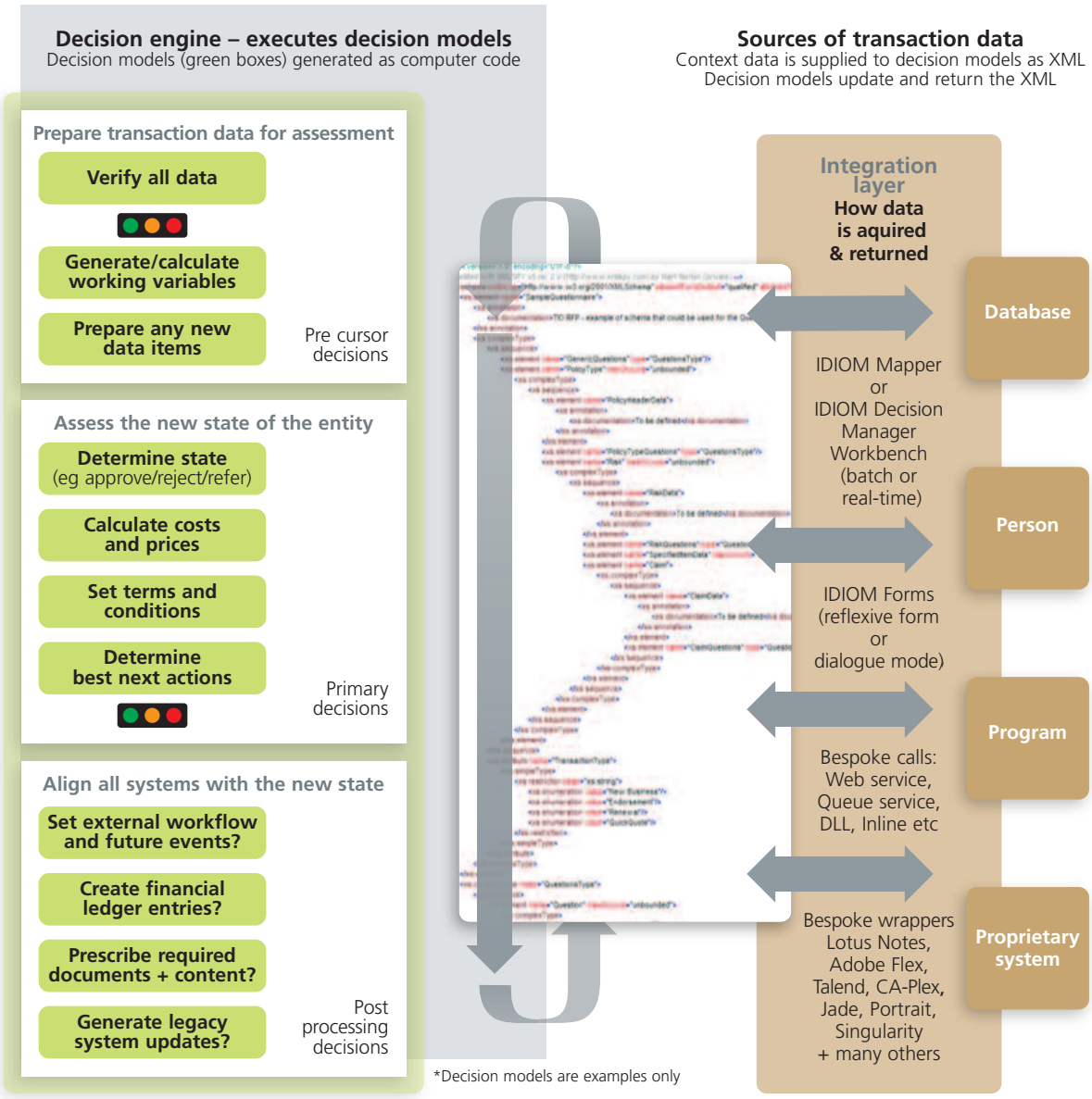
A high level view of the IDIOM approach:

This first of two diagrams describes the policy development life-cycle, including development and testing of past, present, and future business policy using the IDIOM Decision Manager and the IDIOM Decision Manager Workbench.



From development diagram (pg 3)

Policy development life-cycle – deploying and using the policy



A single deployment manifest

- Business owners supply new decision models as Java and/or C# source code included in a digitally signed manifest.
- The decision models transform the data into the language of the policies & then apply the rules, decision by decision.
- The decisions update the XML context data.

IDIOM decision engines address the BIG business problems – applying your business policies correctly, consistently, completely and very, very quickly!

For example single decision engines for:

- All billing rules for 47 hospitals, 121 outpatient clinics; one million invoices per week
- Period updates and daily audit for one million pension fund members
- Multiple products for global insurer onboarding up to 42 million policies per day
- Payroll calculations applying decades of award variations for thousands of employees
- Lifetime clinical pathway
- Full claims adjudication pathway for multiple insurance products

The development and testing of decision models that codify business policy is a *nimble, continuous, perpetual* process that is performed by business 'subject matter experts' (SMEs) using the **IDIOM Decision Manager** – hands-on!
SMEs can also use the **IDIOM Decision Manager Workbench** to support this nimble, continuous, perpetual process with regular, production scale regression tests and simulations to confirm the efficacy of the evolving business policies in their codified form.
The Workbench provides the ability to audit, and if needs be to remediate, existing implemented policy of any complexity and on any scale – actual examples include decades of payroll and pension entitlement calculations. At the same time we can look into the future, using the Workbench to perform policy simulations and what-if analysis to assess and verify the effects of proposed new policy implementations.

The IDIOM approach and toolset become part of the policy development cycle itself.

Customers can use the IDIOM Decision Manager and the Workbench as *policy prototyping* tools to develop and prove new business policies, which they subsequently translate into formal policy narratives. IDIOM generated 'logical English' can be inserted into the policy narrative as an explicit and exact specification of the policy as it *will be* implemented. There are no 'fingerprints' between this logical English and the executable code – so that the business policy and computer systems are *exactly aligned* when implemented.

The second diagram outlines how the business policy can be deployed and executed using an IDIOM supplied decision engine.

The decision engine is primarily an interface (provided in source code form for peace of mind) that executes decision models generated by the **IDIOM Decision Manager** (also in source code form). At execution time, the compiled JARs or dotNET assemblies for the decision models are executed behind the 'decision engine' interface.
The decision models shown in the diagram are examples for illustration only; decision models can address substantially complex business problems. In more mature environments, decision models are often *reused* across related problem domains, giving rise to the multi-modelling concept implied by the illustration. In response to this, IDIOM evolved the 'control model', which is a special instance of decision model that is used to orchestrate multiple (other) decision models at execution time.

The grey arrow linking the two diagrams is used to indicate the flow of decision models from development to production.

Using the IDIOM tools, these models are exported in a comprehensive, digitally signed manifest that includes:

- generated source code for each model;
- compiled code;
- exact descriptions of the models in both XML and 'logical English' formats;
- change logs;
- dependency tracking to other requirements source documents; etc.

It is this manifest which becomes the ultimate 'source of truth' for the business and its policies; as such, it is usually fully integrated into the organization's source control regime, at which time the handover from business to IT control is complete and assured.

Introducing the business transaction

With the above as context, we can now introduce the business transaction. A decision is the business logic behind a business transaction. A business transaction takes a business entity from one state to another (e.g. take a claim from received to approved).
It takes event and context data, applies policy-defined business logic/rules, transforms and stores the data in its new state, and triggers all 'best, next' processes.
We call this transaction a business transaction to convey the idea that the transaction is not complete until the event has been dealt with end to end, traversing all the different data assets and different areas of subject expertise in the application of business policies, and concluding by triggering all downstream processes on a 'best, next' basis. Some of these business transactions are simple and brief, and others are long and complex.
A differentiating point of the IDIOM approach is that it focuses on the complete business transaction. Or it can just focus on one slice of the transaction if that is what is missing. It does not consider any particular slice more important than any other slice, so that it can work by subtraction - taking the whole, appreciating whatever works effectively, and then filling the gaps by acting as 'smart middleware' to de-stress legacy processes.

1. http://www.brcommunity.com/b326a.php Decisioning: A new approach to Systems Development
2. http://www.modernanalyst.com/Resources/Articles/tabid/115/ID/1354/Requirements-and-the-Beast-of-Complexity.aspx

The IDIOM approach – a chronicle

"Decisioning: a new approach to systems development"[1]

This seminal article (published in the Business Rules Journal, Jan 2007) provided the background and conceptual building blocks for decisioning and decision centric development. It highlighted the importance and benefits of decision analysis to good systems design, and at the same time, noted its absence from the then current development methodologies. It drew important distinctions between decisions and 'business rules', and noted the relatively greater importance of decisions over processes as analysis subjects.
It laid the conceptual foundation for *decisions* (now defined as follows) to be a fundamental unit of requirements specification.

Decision: A single definitive datum that is derived by applying business knowledge to relevant data for the purpose of positively supporting or directing the activity of the business.

According to the article, these decisions are defined in the context of a *decision model*, which is defined as:
Decision Model: An ordered assembly of decisions that creates new and proprietary information to further the mission of the business.

And decisioning itself was defined as:
Decisioning: The systematic discovery, definition, deployment, and execution of automated decision making.

Requirements and the beast of complexity[2]

This popular article (published in Modern Analyst, 2010, ~23,000 downloads as of May 2015) updated the above article, and targeted requirements specifications as a significant weak link in the traditional systems development cycle. It presented a case for a new approach to requirements gathering and specification that is based on analysis of *business policy* to identify core business entities and their respective state changes, which are then captured and described as decision models and ultimately implemented as transactions. From the paper's conclusion:

"This 'decision-centric development approach' . . . focuses on a 'missing link' between business strategy and operational systems – the Decision Model. The Decision Model is a new and important requirements artifact that is accessible and understood by both business and development practitioners, giving rise to a previously unobtainable level of shared understanding that can help bridge the gap between business strategies and the systems that support them. The Decision Model gives the business 'hands-on' control over the definition and implementation of its most critical IP – its decision making know-how."

Decisioning – the next generation of business rules[3]

This article (published in both Modern Analyst and Business Rules Journal, 2013) noted that Charles Forgy, the inventor of the rete algorithm (which powered a generation of business rules engines), listed criteria that more or less excludes 'transactions' as an appropriate use-case for the algorithm. Notwithstanding, rete aligned, constraints based rules engines continue to be used in transactional systems, adding significant cost and complexity for little value.
This article then expanded the scope of IDIOM's business rules/decisioning concepts to include the ability to dynamically create data that is aligned with the idiom[4] of the business; that is, with the nouns and noun clauses of the proprietary language (the idiom) that we asserted is always present when describing business rules. By definition, the idiom is always proprietary to the author of the business rules because it is the need to define the proprietary business rules that gives rise to the idiom.
Dynamic transformation from raw data to data that is compliant with the business idiom is needed because a) it is rare that the real-world data is already in the required state for adjudication by the rules, and b) the business idiom is fluid and ever changing – it is the essence of business rules, changing whenever the business changes (c.f. the relatively static nature of the real world data definitions).
The article also highlighted the separation of the doing and decisioning components within a system:

"Service oriented architectures are inexorably trending towards separation of 'doing' components from 'deciding' components. The 'doing' components can be commoditized and outsourced. The truly proprietary code, the code that captures business knowledge and IP, is being increasingly concentrated in the 'deciding' components."

This observation is expanded upon in the more recent Taming the Beast Article[5] described on page 7.

3. http://www.modernanalyst.com/Resources/Articles/tabid/115/ID/2713/Decisioning-the-next-generation-of-Business-Rules.aspx
4. http://www.thefreedictionary.com/idiom [a. 'A specialized vocabulary used by a group of people']
5. See, 'Taming the IT Beast with Decision Centric Processes' http://www.modernanalyst.com/Resources/Articles/tabid/115/ID/3291/Taming-the-IT-Beast-with-Decision-Centric-Processes.aspx

The Role of SQL in Decision Centric Processing[6]

The logical disconnect described in the above paper, between the original design intent of the technology (rete in that case) and its subsequent use, is echoed with SQL. This recent article (published in Modern Analyst (2014) and the Business Rules Journal (2015)) noted that SQL was designed from the outset to address a specific class of problem – set processing.
This class of problem does not fit easily with transactions, and is to a large extent redundant if the data is hierarchically structured around core business entities (and is stored as such). In essence, the article made the observation that the ubiquitous SQL 'join' is fundamentally flawed when used in a transactional context: dependent tables should be joined onto their parent entity; the parent entity should not be joined on to the dependent tables. Again, and in apparent contrast with its original purpose, complex 'join' based SQL is routinely used to underpin modern day transactions.
And to close the circle on the business idiom described in the Decisioning paper above, we observed that the labels that are used to reference data in the business idiom must describe that data in context; that is, the *labels* used for the data values must include the context qualifiers. If we are going to be able to implement business rules that are described using the idiom, then it is necessary that we also describe the data in terms that align with the idiom, so that the rules of data context are matched with the vocabulary of the business idiom and vice versa. This is something not easily achieved using SQL.

This correlation between the business idiom and data context is the essence of the linkage between business policies and their implementation in computer systems. It is also the backbone of business transactions, and in our experience, usually represents the major part of a transaction's internal work effort.

The SQL paper further extended the scope of IDIOM's business rules/decisioning concepts by highlighting data context as an integral part of a decisioning 'requirements specification'.
With a decision authoring tool that understands context, we can quickly and completely define decision models that address the full scope of policy driven decision-making, including validation, transformation, calculation, adjudication and workflow; and which requires only 'simple SQL' (i.e. no joins) to provide complete and error free business transactions of any size and complexity.
And these transactions will often execute much faster than solutions developed using more traditional approaches to use of SQL.
This performance gain is primarily a reflection of the way that SQL is used to support the transactions.

Taming the IT Beast with Decision Centric Processes[7]

This most recent article builds on the earlier articles to address the *process* perspective of the approach. It describes a process architecture and a matching development approach for decision centric systems. It uses 'business transactions' as the core application building blocks.
A business transaction is defined as:

The activity that is initiated by an event that changes the state of a business entity, and which when complete leaves all systems compliant with all business policies relevant for that entity in that state.

A business transaction embodies the complete life cycle of core business entities, and is called and executed for each and every event that affects the entity.
The process complexity is reduced to a finite and known set of *business policies* in the decision models, and a finite and known set of *activities* (usually implemented as services) in the underlying application. Process orchestration is achieved by 'just-in-time', policy-derived decision making to determine the 'best-next' activity or process. Each process or activity only needs to calculate the 'best-next' step to ensure a complete and consistent overall process; the only time best-next can be assessed is *right now*, using the current context and event data to achieve immediate, realtime, process orchestration. We call this approach to process management 'binary BPM'.
The benefits are substantial. The architecture and its associated development approach dramatically reduce the number of moving parts in the system, leading to an equally dramatic reduction in the quantity of bespoke coding required. Throughout the article there are observations regarding the simplification of systems development and execution that are attributable to the approach.
When the approach is used to address complex business transaction processing, the demonstrated results include substantial improvements in the key development metrics of time, cost, and risk. These already encouraging results are accompanied by improvements in business agility, better alignment between business strategy and operational systems, and systems that are more transparent and more durable.

In the approach described by the article, SMEs assume hands-on responsibility for the definition, capture, and deployment of the business policies that manage the entity life-cycles within the business transactions. By using the IDIOM tools to implement the approach, these business SMEs are empowered to take full and complete ownership of the implementation of business policy within the operational business systems.

6. See, 'The Role of SQL in Decision Centric Processing' http://www.modernanalyst.com/Resources/Articles/tabid/115/ID/3109/The-Role-of-SQL-in-Decision-Centric-Processes.aspx'
7. See, 'Taming the IT Beast with Decision Centric Processes'

Making it work with IDIOM

IDIOM tools

As you might expect, IDIOM has been fine-tuning its flagship IDIOM Decision Manager product since its launch in 2001 to fully support the concepts outlined in our series of articles. In fact, the name of our company was an early reflection of the importance of the business 'idiom', and its relationship to both business rules and data.
The 'idiom' is the proprietary language that is used to describe the essence of any business – it is used to describe its business policies, its decision making, and its business rules. And in so doing, the idiom refers to the data in the context required by the rules.
The challenge lies in mirroring this context within the various executables that must implement it.

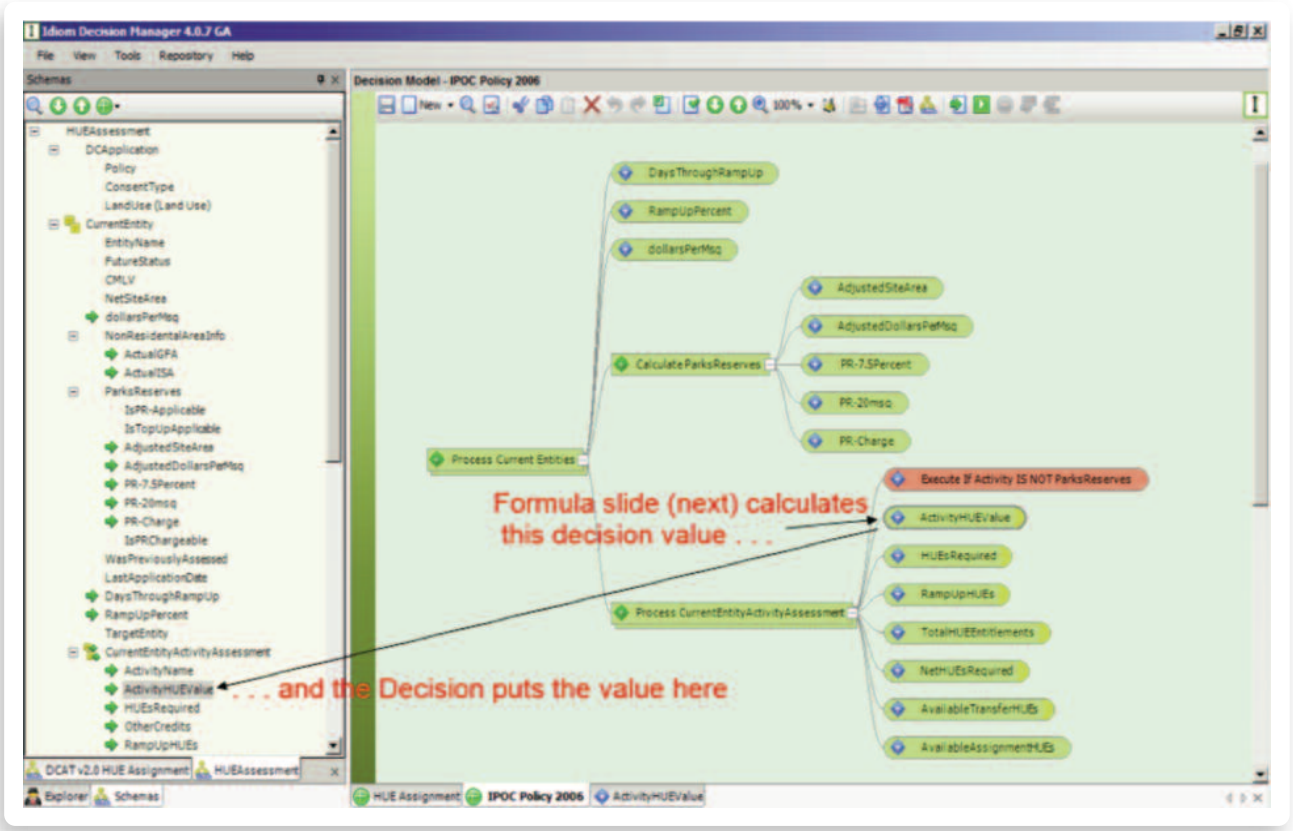
Defining, capturing, validating, & automating the business 'idiom' is the essence of our business and of our flagship product IDIOM Decision Manager.

Furthermore, the high performance relational to XML mapping tool alluded to in the Taming the Beast Article[8] exists as the **IDIOM Mapper**, and is in part responsible for the high performance of many of our applications.
The entire Mapper cycle is also embedded into a comprehensive management application called the **IDIOM Decision Manager Workbench**. The Workbench is an industrial scale, ready-made batch application processor for large scale, high performance batch processing.
A transaction often requires dialog with a human actor. A dialog is easily managed by building an **IDIOM Form** over the business transaction's entity and event data – the same data that is available to the decision models that interpret and maintain it.
An IDIOM form includes the ability to execute IDIOM decision models inside the form on a large scale, field by field if necessary, to ensure a fully reflexive user experience.
Finally, the transaction end state might need to be reported via one or more business documents. The **IDIOM Document Generator** can be used to generate complex business documents under the control of decision models, using only Word authored text and images as additional design input.

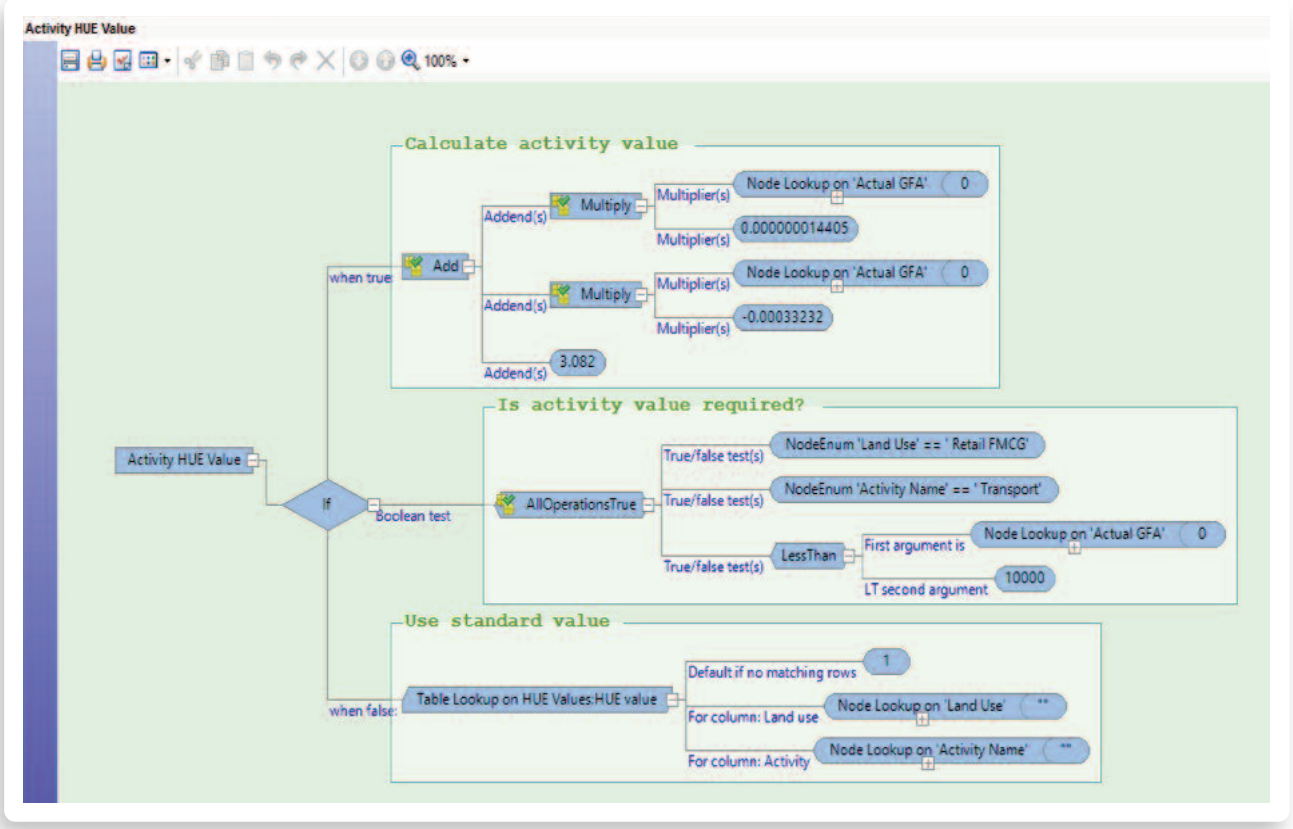
IDIOM Decision Manager

- IDIOM Decision Manager is a tool for graphically modeling, testing, and deploying business decision-making logic – without programming!
- A tool for the policy maker, not the programmer.
- IDIOM automates complex decision-making at the enterprise level, deployable as industrial strength stand-alone components.
- In day-to-day practice it is used by SMEs, or analysts working closely with them.
- Together they model the business policies in terms of both data and decisions (see Decision Model below) before moving on to define the underlying logic that binds them together (see Formula below).
- The decision logic and data are usually modeled together in a single combined process of analysis and definition.
- The data model and the decision model share the same 'context awareness', with current-context and context boundaries visually highlighted at all times within the development palettes.
- Testing of the models is available at all times within the development palettes themselves; full regression testing (incl. 'model answer' differencing) is available in the on-board 'Test Executive' (not shown).
- Deployment as 100% generated software components is fully automated and 'without fingerprints'.
- Example below is a small but real model drawn from a city council implementation of policy that calculates financial contributions to be paid by property developers.
- The problem domain is decomposed using a 'mind mapping' approach until we reach the atomic units that we call decisions (rounded boxes).
- This 'decision model' and the adjacent data model (left hand panel below) are demonstrably aligned and integrated through shared context – validating and strengthening both.
- The data model defines the entity at rest; the decision model defines the valid state transitions. Together they completely define the entity life-cycle and required business policy.
- The atomic 'decisions' provide an easy entry point for specification of the underlying rules details via the Formulas (see next).
- The underlying rules details are easily captured using a 'Lego like' drag-and-drop approach that is 'more fun than playing golf' according to the CEO of one of our largest customers – there is no scripting or coding required to build formulas.
- The rules can be tested immediately within the IDIOM Decision Manager palettes.
- When finished, IDIOM Decision Manager generates computer source code (C# or Java) with a single button click, to be called by any application at run-time using any of a wide variety of supplied interfaces and wrappers (in-line, dll, web service, queue service, and many more).
- And at the same time, it generates the models into business readable PDF documentation.

IDIOM Decision Manager – Decision palette showing a Decision Model



IDIOM Decision Manager – Formula palette showing a formula



Key points of difference

- IDIOM's decision models do for decisions what data models do for data – a powerful abstraction that makes the underlying complexity visible and manageable.
- The models allow data transformations and more traditional business rules to be intermingled. Business rules acting alone are severely limited in their ability to fully resolve complex commercial problems – invariably, in-line data transformations are necessary to facilitate the calculate/adjudicate/act behavior of business rules.
- Context is continuously displayed and actively managed.
- Decision models that incorporate both data and rules behavior enable a further critical capability that is unique to IDIOM Decision Manager – the models can be fully tested using real-world cases directly in the builder palettes. No external technology or application support is required to empirically prove the completeness, consistency, and correctness of the models.

Key points of innovation

- Fundamental redesign of the traditional SDLC by fully separating the development and automation of business policy (deciding) from development of the system's activities that support it (doing).
- Use of IDIOM is effective in spawning a 'Business Policy Development Life Cycle' that is managed independently of and alongside the traditional System Development Life Cycle.

Key value propositions

- 100% alignment of systems based decision making with business policy, because the business owners have hands-on custody and control of the policy definitions actually used by the system.
- Increased agility with reduced business risk through business modeling and empirical testing of policy definitions prior to automated generation and implementation.
- Significant reduction in the business cost of developing and implementing automated business policy.
- Further reduction in software development cost, time, & risk through reduced system complexity, fewer moving parts, & clear separation of concerns.

Further benefits

- Full auditability of policy changes, and visibility of policy implementation through the graphical models and logical English PDF generation.
- Decision model artefacts can be traced to/from 'sources of truth' in underlying business policy documents (Word, Excel) using the IDIOM Tracker for requirements traceability.
- The IDIOM Decision Manager Workbench is available to experiment with and further develop policy independently of the underlying systems and of the SDLC.
- Automated, robust, industrial strength deployment on any scale that can be supported by the 'host application' and its underlying platform.
- Simple injection into legacy systems leading to eventual legacy replacement.

IDIOM Forms

IDIOM Forms is a tool for generating web2.0 forms that embed the power of IDIOM decision models into the form itself. As the user navigates through an IDIOM Form, the integrated decision models can be executed on an element by element basis to make various business calculations and assessments; and to modify the form's meta-data. Control of the form's meta-data allows the decision models to dynamically adjust the visible shape and content of the form on an element by element basis.
A single IDIOM Form can be used to process complex business transactions (e.g. an insurance application, a clinical pathway, a loan application) through to closure, including such functions as validation, transaction acceptance, costing or pricing, up-selling, and determination of subsequent workflow amongst others. Usually, a single form is used to manage the full life-cycle of the subject entity (i.e. not just individual state changes).
The IDIOM Forms Engine (the runtime component of IDIOM Forms) is production hardened after many years use on thousands of servers throughout the NZ health sector. The deployed IDIOM Forms Engine became the basis for the NZ Health Information Standards Organization (HISO) forms standard and is currently the most compliant under that standard. The Forms Engine is now also widely used in the finance sector for insurance underwriting and claims management.

IDIOM Decision Tracker

The IDIOM Decision Tracker is a tool to map Microsoft Word and Excel policy documents to IDIOM decision models for full bi-directional traceability between corporate policy definitions in the Microsoft documents and their actual implementation as IDIOM generated decision engines.

IDIOM Mapper

The IDIOM Mapper builds an in memory XML document that mirrors the database structure that it is reading from. It generates 'simple SQL' from an XML configuration document, and then executes a full round trip transaction cycle: from database to XML; then rules execution (multiple decision models); and from XML back to the database if required. All done extremely quickly, either in single transactions or in batch. The Mapper process is thread-safe and can execute in many process streams for scalability that is limited only by the database and its platform.

IDIOM Decision Manager Workbench

The IDIOM Decision Manager Workbench is a business user operable application for running decision models across enterprise databases on a large scale without the need for IT technical support.

- A generic batch processing platform for use by IT and/or business operations.
- A platform for the auditor, the business policy manager, the product manager, the corporate business strategy manager and the IT manager.
- An audit tool for identifying, reporting, and managing anomalies, errors, and issues of concern in any database.
- A simulation tool for comparing the performance of current and 'to be' versions of any automated policy.
- A tool that can intelligently modify the inbound data to simulate changes in the make-up of inbound transactions over time.
- A data masking tool, able to replicate large scale databases with all personally identifying details masked from its own library of several million fake identities. Relationships between identities in the source data can be maintained, notwithstanding the use of fake identities.
- A conversion tool: able to read data from one system in its proprietary format; intelligently transform it through one or more decision models; and then output it to a new system also in its proprietary format. Millions of entities described by hundreds of tables can be supported.

Examples of use

Superannuation Fund Administrator: Perform all period end processing, including fee's, insurances, member adjustments, entitlements, and reporting, for several hundred thousand fund members.

Insurer: Compare the current underwriting policy with a proposed underwriting policy across a recent portfolio of 500,000 insurance policies to determine potential changes in the rate of referral and its attendant costs.

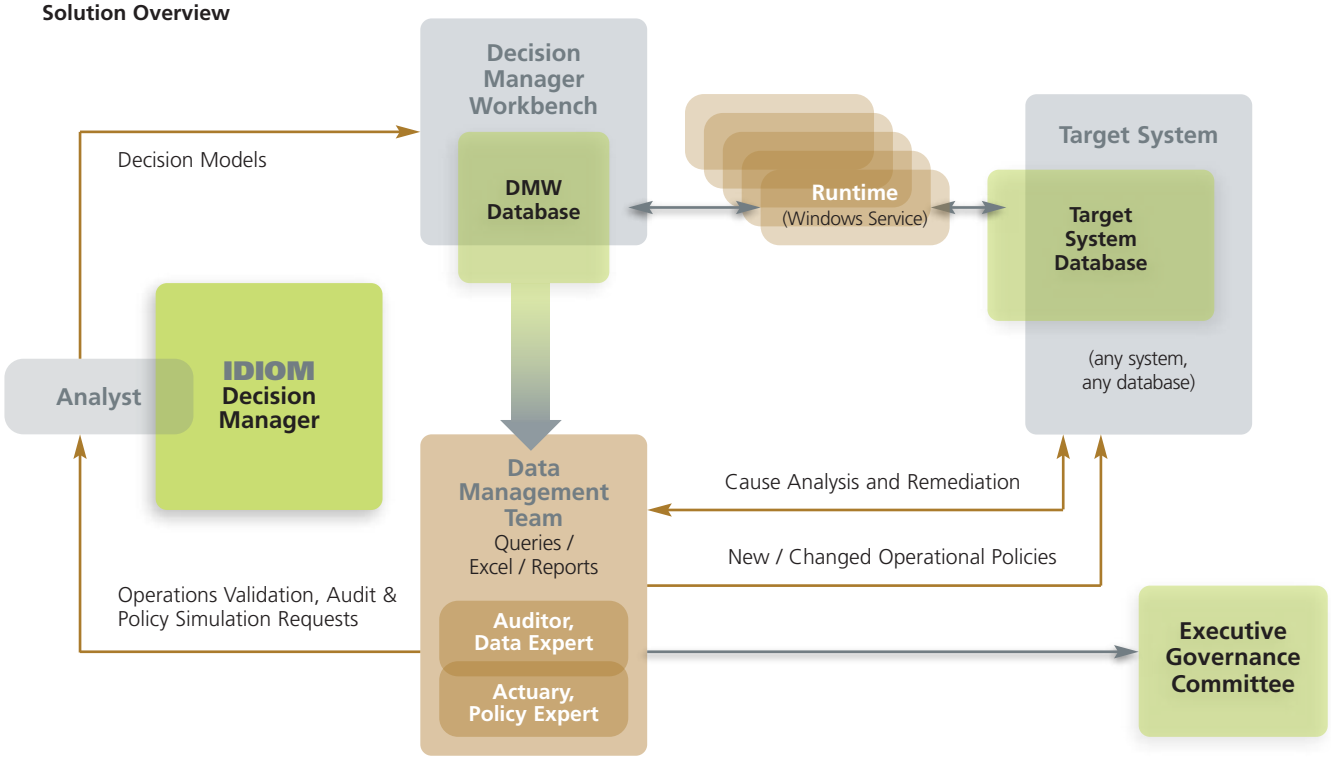
Superannuation Fund Administrator: Run 100's of distinct audit tests across 1million member accounts on a daily basis to independently verify the production data.

City Council: Compare this year's rating policy with next year's proposed rating policy for each property in a city of more than 500,000 ratepayers to identify outlier changes in the rates actually charged.

Insurer: Dynamically modify key attributes of real transactions to simulate changes in the make–up of in-bound business, and assess the impact of these changes across an entire portfolio.

IDIOM Data Management

Solution Overview



Enterprise class features

- Full authorization and audit controls down to the field level for all users of the platform.
- Seamless operation across multiple user definable environments, for instance Development, UAT, Simulation, Production.
- High performance, including parallel processing on a large scale across multiple machines, for instance running up to 24 Processes on an i7 class CPU, with multiple CPU's possible, each logging back to the Workbench for centralized management.

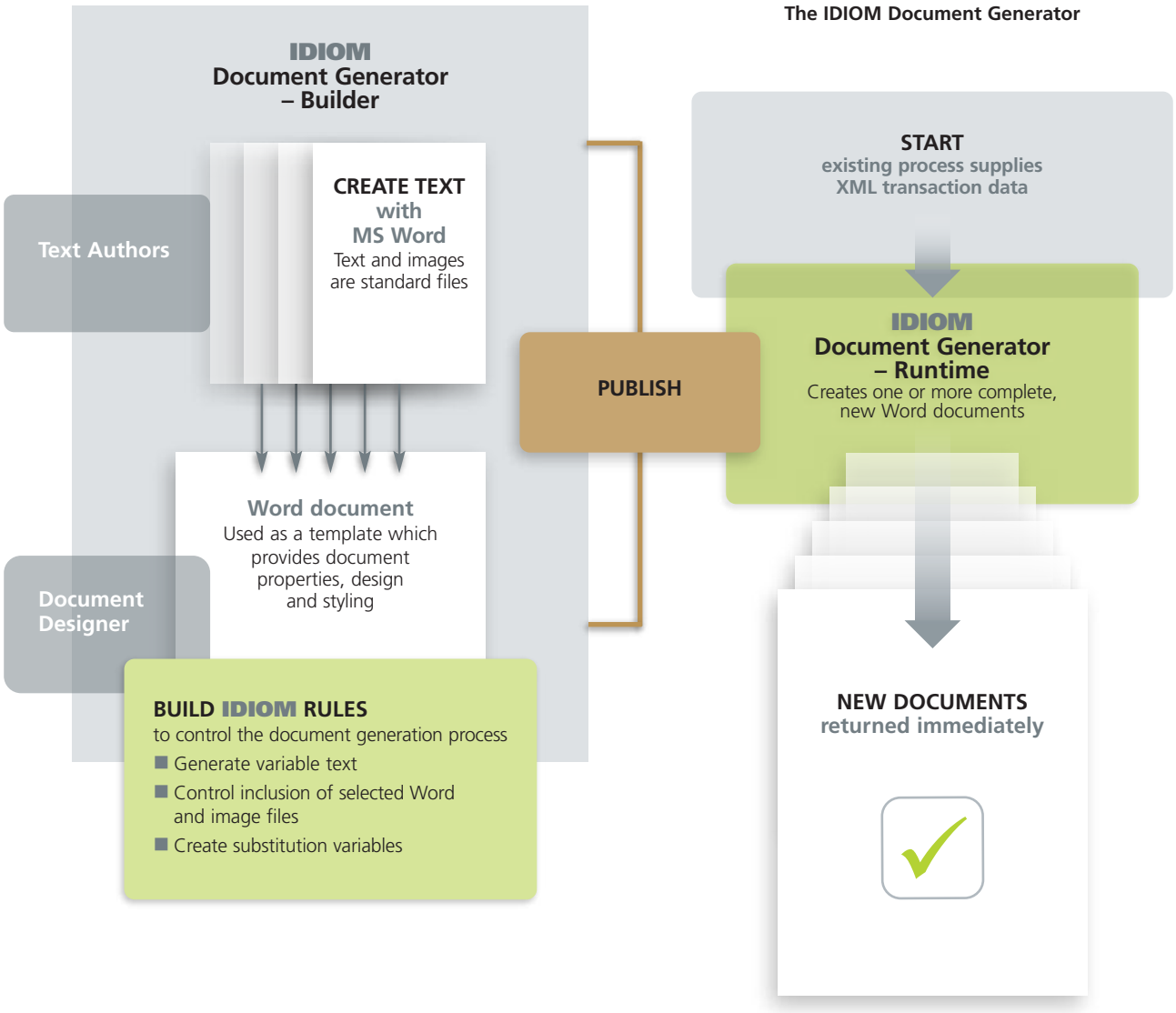
Measured performance

- Daily pass of 1million pension fund members each comprised of a join of 20+ complex member related tables (~one billion rows in total).
- 10's of decision models implementing hundreds of individual member tests.
- Alerts captured and reported daily for start of business.
- Run daily in 2 hours using one i7 class processor, with data drawn from an IBM iSeries.

IDIOM Document Generator

The IDIOM Document Generator is a tool to collect and collate Word documents (which are used as document 'templates') and many associated Word text fragments ('blurbs'), and to assemble these into complete documents under the control of a decision model.

At document design time, the Document Generator inspects the templates and blurbs and builds a list of all blurbs (including nested blurbs) that could be used for each candidate document; and it also builds a list of all of the variables declared within them. It then places these two lists (blurbs and variables) into an XML 'control document'.



A decision model is then built that will extend this XML control document at runtime when the specific business entity context is known. This decision model will analyse the business entity context data in order to set flags to control the insertion (or not) of each blurb; and it will derive the substitution values for each of the listed variables.
At runtime, when the actual business entity data is provided and the decision model has executed, the Document Generator will read the tags to insert the desired blurbs into the template, followed by substitution of all variables, so that large and complex documents can be generated automatically under the control of the decision model to reflect the exact circumstances of the underlying business entity.
The Document Generator also allows for insertion of images and for calling bespoke programs to insert additional generated text or images. Documents can be generated into a number of formats, including Word and PDF. In day to day use by SME's, the Document Generator only requires standard Microsoft Word and IDIOM Decision Manager authoring skills to define new documents.

Putting the approach and the tools together

Using the approach

We have highlighted the role of the business 'idiom' as the language of business policy, and the importance of its automated equivalent in a decision centric system. In order to automate the idiom, we need to view data in context. We have also described how the XML Schema provides us with a partial 'context map'.
The IDIOM approach uses the IDIOM Decision Manager to provide the SMEs – the subject matter experts who actually define the decision models in accordance with business policy – with a drag and drop GUI to graphically build decision models, including the validations, transformations, adjudications, calculations, and workflow logic (collectively, the decisions) directly over the XML Schema without needing to know any XML navigation syntax or other programming skills.
The schema provides our basic context map, so that the complex task of understanding and controlling context is largely managed by the tool. The SME is only able to reference data that are valid for the current context, which is the locus of the decision that is currently in focus, and is always a single, unique point on the schema.
The decisions and their related logic are graphically captured using the IDIOM Decision Manager's decision model and formula palettes.

A decision model is strictly hierarchical, executing left-to-right/top-to-bottom so that the decision execution sequence is graphically displayed and easily controlled.

The IDIOM Decision Manager allows the SME to dynamically create new (temporary) data at any time. These plug into the schema as required, and are usable in the same way as any other value.
The IDIOM Decision Manager also allows the SME to create test data and to test any part/whole/group of decisions directly in the development palettes during the development process, with context being graphically displayed as the test execution takes place. Real XML entity records from operational business systems can also be used immediately and directly for this purpose.
When finished, entire libraries of test cases can be regression tested within the IDIOM Decision Manager, before generating the source code and deploying it with a single click for system testing. Again, this can be a use for the real-world XML entity records described above.
Because the decision models and their associated schemas provide a complete specification, a high performance and well documented computer program can be completely and automatically generated 'without fingerprints'.

The following outline summarizes this new decision centric approach:

1. Develop an XML Schema to describe the 'real-world' entity that is the subject of the transaction, to be used as a 'context map' for subsequent decision model development.
2. If an existing database exists, use the IDIOM Mapper to generate and execute a series of 'simple' SQL queries to collate a complete, in memory XML object as defined by the schema. The reverse mapping can also be generated for write back if required.
   Or, you can use XML entity records directly, either from the file system or a database. The database design described in the '*Simplified Data Model*' section of the Taming the Beast Article[9] supports an efficient and simple hybrid relational/XML approach.
3. Use IDIOM Decision Manager's drag and drop GUI to declaratively build out the decision model(s), using the schema defined 'context map' as an active and extensible requirements template.
4. Test the evolving transaction decision model 'in situ' to ensure completeness, consistency, and correctness at all times.
5. Regression test in the same tool to confirm absence of unintended consequences; and/or run simulations to verify that changes in the underlying business policy achieve the desired policy objectives.
6. Generate and deploy without further manual intervention.

9. See, 'Taming the IT Beast with Decision Centric Processes'

Further effects of the approach

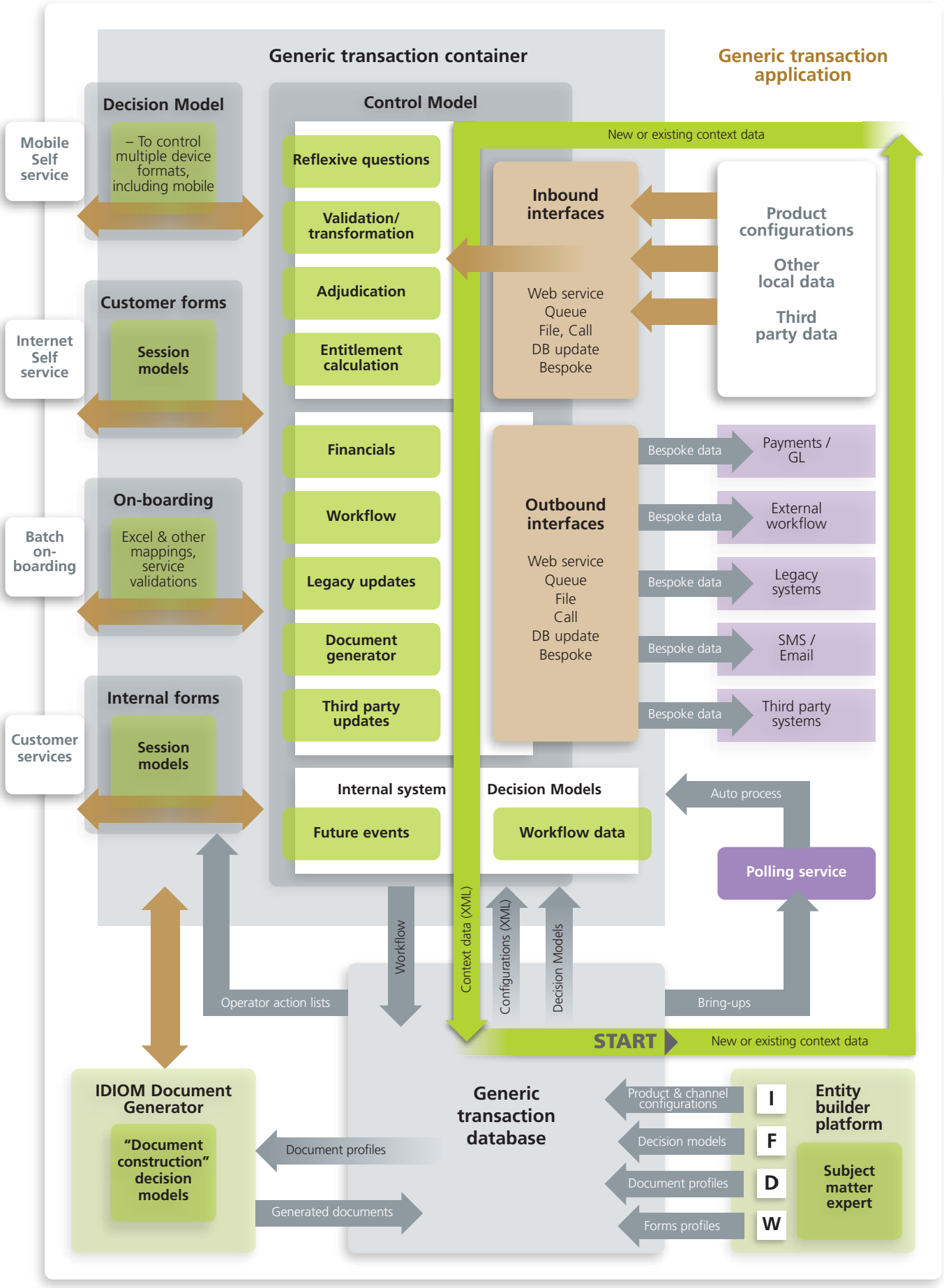
Building a complex transaction using the above approach has many positive downstream effects.

1. The 'decision model' is tightly structured, and able to be rendered into many formats, including logical English, computer source code, and XML. It is therefore transparent, auditable, and reliable as a specification of the executable code.
2. This is a tool for the SME. The high degree of automated assistance and reduced development complexity means that the SME is only contributing what they already know – their own proprietary business knowledge. Productivity is quickly (days, not weeks) and reliably achieved; arcane IT skills are not needed and offer no advantage.
3. Generation of code means that the code per se does not need to be tested; only the logic supplied by the SME needs to be tested. This means a substantial reduction in testing effort.
4. The automated tool assistance and reduced human input reduces the potential for errors. This gain is compounded by the extensive on board testing support already described, so that it is rare for a decision model to be anything other than complete, consistent, and correct when released into later-stage testing cycles.
5. All aspects of the decision model support continuous, perpetual versioning. When the models are implemented as 'content' in computer systems, they provide a practical and robust mechanism for SMEs to independently develop, manage, and deploy the organization's codified knowledge (including complex transformation, adjudication, calculation, and workflow policies) on a daily basis – continuously and perpetually.
6. The decision models are technology agnostic and technology independent, forming a complete historical record and the ultimate 'source of truth' for the organization's proprietary knowledge; extracted, tested, confirmed, and documented by that organization's SMEs in the normal course of business.
7. A tool assisted approach as described allows SME's to capture, test, and deploy this knowledge extremely quickly. In a 100 developer year project, 80% of the system code was generated from decision models that were built in less than 20% of the total development hours – all of which were contributed by business analysts drawn from business branches (i.e. not IT specialists).
8. This development efficiency is sustainable over the long term, offering huge business agility with reduced cost, time, and risk
9. A decision model is a durable life-long artefact that defines the business in perpetuity. As a complete record of an organization's knowledge, and with multiple and extensible source code generation options, there should never be another 'legacy system'.

The generic transaction

The following diagram shows a pro-forma transaction that is capable of addressing many standard entity life-cycles, including those that we have used as examples in this document (insurance policies, claims, loans, patient episodes, patient clinical pathways, frequent flyers, property rating, entitlements, et al).

The inherent variations in the underlying entities, and by extension, their life-cycles, are all addressed by the IDIOM tools within the 'Entity Builder Platform'. The SME has extensive control over the definition of these entities, and of the actions that form their respective life-cycles. Provided that the various service end-points are available to supply data to the transaction on request, and to respond to the updates generated by the transaction on termination, then the transaction itself can be a black box. For instance, a type ahead address lookup to verify property details could equally be serving an insurance policy, a claim, or a loan; it is immaterial to the service which of these it is.
The majority of the development effort in building the transaction is in the decision models, which are compiled and executed behind the standard interface that is built into the generic business transaction itself. Independently measured projects have assessed this declarative development effort as being at least 20 times more efficient than traditional SDLC approaches to achieve the same end.
The generic business transaction container would use the event context to look up and load the relevant decision models based on the nature of the event that it is responding to.
An IDIOM solution could also use the IDIOM Forms engine to run the web forms for the users of the system (customers or back-office). And the Document Generator could be deployed as a service within the host environment to generate and serve business documents. The runtime components of the IDIOM Forms and the IDIOM Document Generator are included in the diagram.
Note that IDIOM Forms and IDIOM Document Generator both use IDIOM Decision Models to drive their respective runtime meta-data, which is what makes these runtime engines truly generic.
The ongoing development of the decision models, forms, and documents requires the builder versions of each of these tools.
This is shown inside the Entity Builder Platform. The Workbench is also available in this platform to assist with policy modelling on the one hand, and regression testing on the other.
This entire transaction can be built with only a handful of bespoke code, much of which is available in IDIOM's existing supplied libraries.
The web services, shown as the other integration end-points, would need to be built bespoke according to the requirements of the other parties; this often constitutes the largest bespoke coding effort implied by the diagram.


Generic transaction container
Generic transaction application
Decision Model
– To control multiple device formats, including mobile
Mobile Self service
Control Model
Reflexive questions
Validation/ transformation
Adjudication
Entitlement calculation
New or existing context data
Inbound interfaces
Web service
Queue
File, Call
DB update
Bespoke
Product configurations
Other local data
Third party data
Customer forms
Session models
Internet Self service
Financials
Workflow
Legacy updates
Document generator
Third party updates
Outbound interfaces
Web service
Queue
File
Call
DB update
Bespoke
Bespoke data
Payments / GL
External workflow
Legacy systems
SMS / Email
Third party systems
On-boarding
Excel & other mappings, service validations
Batch on-boarding
Internal forms
Session models
Customer services
Internal system
Future events
Decision Models
Workflow data
Auto process
Polling service
Workflow
Context data (XML)
Configurations (XML)
Decision Models
Operator action lists
Bring-ups
START
New or existing context data
IDIOM Document Generator
"Document construction" decision models
Document profiles
Generated documents
Generic transaction database
Product & channel configurations
Decision models
Document profiles
Forms profiles
I
F
D
W
Entity builder platform
Subject matter expert

Versioning and release

Continuous, business led development

One of the important objectives of a transaction/decision centric application is continuous development and release of business policy by business owners, or more specifically, the SMEs. This business policy is usually focused on management of entity life-cycles, whether as formally defined 'business products' or as entity "case management" or other services.
Policy adjustments should be able to be developed, tested and deployed at any time. The intent is not to actually deploy changes (say) daily, but to be able to deploy new policy at any time the business requires, without the delay and risk inherent in traditional 'code-drops'.
All parts of the business transaction, including all policy changes, are effective dated and cumulative forever. The business transaction is always able to operate 'as at' any prior date. The IDIOM approach even allows effective dating that spans changes in the entity schemas.

Continuous development and deployment is helped by IDIOM's zero script, fully declarative approach, coupled with a strong focus on versioning of and within all of the IDIOM components, so that continuous, rapid change can be sustained – perpetually.

With a transaction/decision centric application there is a new dichotomy in versioning: the schemas, forms, rules, and business documents, and all of their associated reference data, can all be deployed externally to, and independently of, the code-base of the host application. There are now two distinct classes of versioning – changes that are applied as new declarative content (in the business transaction layer); and those that require system code changes in the host application layer, via a traditional 'code drop'.
Broadly speaking, the former can be managed by suitably trained business SME's and policy experts using the new "business policy life cycle", while the latter follows established IT systems development best practice.
While the traditionally expensive IT development practice is still critical in a decision centric application, it is now dealing with a much smaller part of the total installed code-base – in one carefully measured 'green-fields' project that developed a full insurance system, the native coded 'host application' was less than 20% of the entire run-time image of millions of lines of code. The rest was generated from drag and drop designs produced by SME's using an independent development cycle that delivered functionality 20 times faster than traditional code development[11]. This is one reason for the dramatic improvement in time, cost, and risk that is achievable with decision centric development.

The continuous versioning that is inherent in the IDIOM approach is designed to support nimble continuous perpetual change in a decision centric application.

Transaction versions

A 'transaction version' is the net effect of all of the rules that implement the entirety of the business policy as it was defined at the point in time that is defined by the transaction's effective date. Transaction versioning is therefore implicitly an effective dating regime.
Each change in the rules or of any other aspect of the decision model within a business transaction implies a version update for that transaction, and by extension, for the product or service that it supports. All such versions always remain valid and operational, subject to the effective date of the transaction at runtime.
Business transactions can also be versioned on criteria other than effective date through configuration or rules changes – for instance, the term for specific conditions might be negotiated on a customer by customer basis, so that each customer might see a different version of the same transaction rules on the same date. This is achieved by dynamically setting the effective date using rules that apply the specific conditions.

Effective dating

Effective dating is fully supported by the IDIOM Decision Manager.

An effective date is always present during the execution of decision models, which may be set by default (i.e. today); or by the calling application; or by rules within the model itself (it is possible to change the effective date during the processing of a decision model using business rules).
The effective date that is set by/for the transaction as above is used to automatically select every aspect of the decision model at runtime. All aspects of the decision model, from the complete model itself down to single reference-data values and/or rules, are start and end dated as required (usually automatically).
Effective dating is not supported by the W3C XML Schema standard, and since XML Schemas are the basis of IDIOM forms, this also limits effective dating in forms. However, effective dating in the schemas and forms can be emulated by the rules that process them. For instance, effective dating the presence of an element in a form is achieved by effective dating the decisions that instantiate that element.
Finally, enumerations (i.e. pick-lists etc.) pose a specific versioning problem in forms. While the Schema standard does support enumerations, these cannot be used because of context and effective dating constraints which are not supported. Therefore, enumerations in an IDIOM form are typically generated by purpose built decision models, so that the enumerations used in the forms are selected at runtime according to both context and effective date.

11. See this paper for details: http://www.idiomsoftware.com/uploads/gallery/temp/1346423022IDIOM_Development_Performance.pdf

Versioning with new data

As we have noted, the business transaction concept is generic. When implemented as described, this means that entirely new business entities and their related meta-data can be introduced at any time without changing the host application code. For the sake of clarity, new types of entities, including new product types, can be introduced and, unless new activities are required,
can be fully functional within the existing host application without any change in that application's bespoke code base.
Existing IDIOM design patterns also provide just-in-time auto-migration of existing data onto new schema designs, which allows versioning of the underlying entity data designs.
This means that multiple schemas, forms and associated decision models, and all versions thereof, need to be managed and available simultaneously within the host application. A database design that can contain and manage all of these artefacts is described in the 'Simplified Data Model' section of the Taming the Beast Article[12].

Real decision centric projects

Following are samples of real decision centric projects that highlight the *significant problem complexity*, the *operational scale*, and the *range of business domains* that the approach is capable of servicing.

Examples of decision model complexity

Two recent projects illustrate the complexity that can be addressed using decision models. Each of these examples was implemented by a single decision model.

Defined benefit scheme

Defined benefit schemes are contracts between a pension fund and its members. They are renowned for their complexity and long-life.
IDIOM has recently codified the trust deed (described in the table below) for one large fund with a membership in the hundreds of thousands.
The underlying trust deed is more than 30 years old, and the complexity of the entitlement calculation has grown considerably since it was first drafted.

- The calculations require large amounts of source data – including 30 plus years of working hours and salary history, with many rules applied at source to adjust for historical data anomalies (e.g. an employment terminated with no record of that employment ever commencing);
- A long standing member can have 40 or more separate periods of employment service, as a new employment service period commences with any change in service: role, employer, payroll, part time work, leave without pay, temporary incapacity, permanent disability, deferral from the scheme, leaving the scheme, re-joining the scheme, etc. with each requiring special treatment in the calculations;
- There are 40 or so intermediate calculated values that contribute to 30 or so benefit entitlement calculations, all of which incorporate many historical changes in the scheme over its 30 year life (e.g. prior to a particular date a calculation is undertaken in a certain manner with a certain set of rates applicable, which is then changed a few years later, etc.), so that some benefit calculations have 4 or 5 of these historical changes inside each calculation method;
- The result is calculations that are multi-layered, starting with several high level components and cascading down through multiple layers of sub-calculations so that some individual calculations have more than 100 discrete sub-calculations within them;
- Each calculation or sub-calculation might need to include indexation by either daily or quarterly CPI in different circumstances – e.g. just for a certain period, or from the date of a certain event forward or backward in time, or for a certain event only when other specific conditions apply;
- Some calculations require dividing period-of-service base data into multiple smaller time-boxes based on externally supplied dates (e.g. Scheme Date, Calculation "as at" Date, 20 year service threshold date, 65th birthday), with different calculation rules applying in each new time-box.

12. See, 'Taming the IT Beast with Decision Centric Processes'

A member's entitlement under the scheme can change on a daily basis; as you can imagine, the current entitlement is keenly sought on a regular basis by the members managing their affairs approaching and post retirement.
Calculation of this entitlement was previously performed in batch by a legacy system that took >48 hours to run; furthermore, these were only top-up calculations for the current period.
The IDIOM version of the calculation re-calculates the full 30+ years so that it can address anomalies in historical data and calculations; it also runs a full order of magnitude faster than the legacy system 'top-up only' equivalent. In fact, this calculation speed, and the fact that each member is processed within a discrete transaction, means that the calculation can now be performed in real-time as well as in batch. The downstream benefits of this are significant.
For instance, the fund administrator can now provide modelling of future changes in real-time so that the member can use the calculation for financial planning purposes; and/or, the fund can provide a service that provides advance information to members regarding future regulatory or tax changes, scheme changes, etc.; and/or the fund can perform financial forecasting and affordability testing for the investment managers of the fund.

Payroll – audit and remediation

The aim of this project for a Government Agency was to:

- Recalculate and remediate termination payments for all terminated employees
- Recalculate and remediate past payments for active employees
- Provide a transaction to correct the current payroll system on a go-forward basis, pending its replacement.

To achieve these aims it was necessary to build an employee level business transaction that:

- Uses the IDIOM Mapper to retrieve the required source data from the payroll system database (e.g. Timesheets, Employment Details, Allowances, Payments Made, Leave Taken, LWOP Days etc.)
- Constructs a temporary data structure suitable for the analysis
- Calculates what the payments should have been, and compares these with payments actually made to calculate remediation amounts
- Produces a report with references to the individual inputs and the intermediate calculated structures to provide a detailed audit trail to support the remediation and ongoing payments.

The complexities involved in this project included:

- ~10,000 members terminated, ~20,000 members current
- Total remediation pay outs of tens of $$million over a 10 year horizon
- Poor discipline and procedures in the underlying payroll systems resulted in varying data quality, including inconsistent and irregular data (e.g. leave taken following termination) that needed to be rectified according to new and agreed remediation policies
- Data structures and database keys changing over time as systems migrated
- Running different calendars for different parts of the country
- Data to be analysed reached as far back as 40 years
- All nuances in the legislation and awards over the full term including definitions of Base Rate, Ordinary Rate and Average Weekly Earnings
- Creating an intermediate data structure representing each day the person was employed, to be marked with the amount the employee was paid that day in aggregate, or if leave was taken
- Calculating various cash-ups: Annual Leave, Long Service Leave, Shift Leave and Statutory Holiday pay outs
- Comparing the Calculated Amount to the Amount Paid and calculating the pay-out amount
- The Amount Paid was not uniformly found and needed to be located from different places depending on time period

Examples of operational scale

The decision centric transaction approach allows applications to scale well as the following examples show:

1. A global leader in micro-financing can on-board up to 42 million new business applications per day on a standard PC. Many decision models, including channel and partner specific models, are used to process each application.
2. The hospital authority of a major economy is using IDIOM decision models to generate 1 million invoices/week across 47 hospitals, 121 out-patients clinics.
3. A pension fund is performing several hundred distinct audits that are described by dozens of 'decision models' for ~1 million fund members daily. The scale of data is substantial, involving a total of ~1billion rows of data that are extracted from a relational database in ~30 minutes using the IDIOM Mapper. Current audit processing takes around 2 hours on a single i7 class processor using 24 processes; this time can be reduced to match the data delivery rate by increasing the number of processors, something that is easy to achieve in a decision centric application.

Examples of projects across various domains

The following list is a snapshot of some of the projects that IDIOM has been working on over the *last 2 years*. This list is provided as a working example of the type and nature of development works being generated by 'decision centric' thinking.

Major Regional and Global Insurer: Conversion of dozens of insurance products to IDIOM Forms and IDIOM Decision Manager, supported by IDIOM Decision Manager Workbench for product modelling and testing. These products are large and complex, with thousands of data fields involved per customer policy application. Some policies cover thousands of individual risks.

Major Regional and Global Insurer: Centralised underwriting and rating service for core insurance products based on IDIOM Decision Manager. This project positions IDIOM as a core technology in one of the largest insurers in the world.

Health Funding Authority: Codify hospital funding rules using IDIOM Decision Manager. All patient episodes state wide are evaluated and costed.

Intercity Bus Operator: Full system build, using IDIOM Decision Manager for the just-in-time pricing that differentiates the operator.

Superannuation Fund: Automation of month end processing – insurances, fees, member adjustments, and reporting using IDIOM Decision Manager. 120,000 members processed each run, with time to run reduced from 2 days to a few hours when compared with the prior legacy system on the same machine.

Fund Manager: Full system build with significant use of IDIOM Decision Manager and IDIOM Mapper.

Insurance Service Provider: A nationwide service linking all life insurers with GP's for collection of medical history data, using IDIOM Forms that manage >1000 data items/form.

National Health Service: Automation of a lifetime clinical pathway using IDIOM Decision Manager.

National Health Service: Automation of organ donor/recipient matching to provide an immediate, fair, and transparent 'next recipient' selection, using IDIOM Decision Manager.

Micro Insurance Provider: Full insurance systems build for multi-country use, using IDIOM Forms and IDIOM Decision Manager. The number of active policy holders is 10's millions.

Award Winning Airline: Frequent flyer loyalty calculations using IDIOM Decision Manager.

Life Insurer: Underwriting management process for enterprise wide use by 100's of underwriters using IDIOM Decision Manager and IDIOM Forms. This application is cloud based, and credited with helping the underwriter go from startup to market leader in 3 years.

Life Insurer: Claims management process using IDIOM Decision Manager and IDIOM Forms. This application is cloud based.

Border Protection Agency: Adjudication of entry pass/fail at border control points using IDIOM Decision Manager.

Regional Local Body (NZ): Web application to calculate developer contributions using IDIOM Decision Manager.

As indicated above, the IDIOM tools provide a cost effective and risk averse means to realize the benefits of decision centric approaches.

Please contact us if you would like to discuss any aspect of this publication, to see a product demonstration, or to contemplate a proof of concept.

We would be happy to oblige.

Mark Norton,
CEO and Founder, IDIOM Limited

About IDIOM

Established in 2001, IDIOM Limited is a private company based in Auckland, New Zealand.
IDIOM develops and licenses decision-making software that automates business policy on a large scale, making systems more transparent, more agile, and more durable, while reducing development cost, time, and risk.

IDIOM's innovative business oriented software is used by business users to graphically define, document, and verify corporate decision-making and related business rules; it then auto-generates these into small footprint, non-intrusive software components for use in systems of any type or scale. IDIOM is a pioneer in the development and use of decision automation concepts, and has applied these concepts to develop and automate business policy for customers around the world in local/state/central government, insurance/superannuation/finance, health admin/clinical health, telecoms, logistics, and utilities.

IDIOM automated business policy and decision making extends far beyond mere business rules, so that larger and more complex decision making can be fully delegated to business experts. IDIOM enabled development and management of policy based decision making by policy owners creates a propitious 'business policy life-cycle' that significantly improves business agility and transparency.

IDIOM develops and licenses: **IDIOM Decision Manager™**, **IDIOM Forms™**, **IDIOM Document Generator™**, **IDIOM Mapper™**, **IDIOM Tracker™**, and **IDIOM Decision Manager Workbench™**.

IDIOM
PURE DECISIONING