Advances in Intelligent Systems and Computing 263

Thouraya Bouabana-Tebibel
Stuart H. Rubin *Editors*

# Integration of Reusable Systems

Springer

# Advances in Intelligent Systems and Computing

Volume 263

*Series editor*

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

For further volumes:
http://www.springer.com/series/11156

*About this Series*

The series "Advances in Intelligent Systems and Computing" contains publications on theory, applications, and design methods of Intelligent Systems and Intelligent Computing. Virtually all disciplines such as engineering, natural sciences, computer and information science, ICT, economics, business, e-commerce, environment, healthcare, life science are covered. The list of topics spans all the areas of modern intelligent systems and computing.

The publications within "Advances in Intelligent Systems and Computing" are primarily textbooks and proceedings of important conferences, symposia and congresses. They cover significant recent developments in the field, both of a foundational and applicable character. An important characteristic feature of the series is the short publication time and world-wide distribution. This permits a rapid and broad dissemination of research results.

*Advisory Board*

Chairman

Nikhil R. Pal, Indian Statistical Institute, Kolkata, India
e-mail: nikhil@isical.ac.in

Members

Emilio S. Corchado, University of Salamanca, Salamanca, Spain
e-mail: escorchado@usal.es

Hani Hagras, University of Essex, Colchester, UK
e-mail: hani@essex.ac.uk

László T. Kóczy, Széchenyi István University, Győr, Hungary
e-mail: koczy@sze.hu

Vladik Kreinovich, University of Texas at El Paso, El Paso, USA
e-mail: vladik@utep.edu

Chin-Teng Lin, National Chiao Tung University, Hsinchu, Taiwan
e-mail: ctlin@mail.nctu.edu.tw

Jie Lu, University of Technology, Sydney, Australia
e-mail: Jie.Lu@uts.edu.au

Patricia Melin, Tijuana Institute of Technology, Tijuana, Mexico
e-mail: epmelin@hafsamx.org

Nadia Nedjah, State University of Rio de Janeiro, Rio de Janeiro, Brazil
e-mail: nadia@eng.uerj.br

Ngoc Thanh Nguyen, Wroclaw University of Technology, Wroclaw, Poland
e-mail: Ngoc-Thanh.Nguyen@pwr.edu.pl

Jun Wang, The Chinese University of Hong Kong, Shatin, Hong Kong
e-mail: jwang@mae.cuhk.edu.hk

Thouraya Bouabana-Tebibel
Stuart H. Rubin
Editors

# Integration of Reusable Systems

Springer

*Editors*
Thouraya Bouabana-Tebibel
Laboratoire de Communication dans les Systèmes Informatiques
Ecole Nationale Supérieure d'Informatique
Algiers
Algeria

Stuart H. Rubin
SPAWAR Systems Center Pacific
San Diego
USA

ISSN 2194-5357 ISSN 2194-5365 (electronic)
ISBN 978-3-319-04716-4 ISBN 978-3-319-04717-1 (eBook)
DOI 10.1007/978-3-319-04717-1
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014931756

© Springer International Publishing Switzerland 2014
This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

# Preface

Software reuse and integration has been described as the process of creating software systems from existing software rather than building software systems from scratch. Whereas reuse solely deals with the artifacts creation, integration focuses on how reusable artifacts interact with the already existing parts of the specified transformation. As a consequence, every integration can be seen as consisting of an analysis of the parts and of their subsequent synthesis into the new whole.

Although significant progress has been made on software reuse and integration, some important issues remain to be fixed. One of these addresses scalability by showing how to make best use of reusable components for very large systems. "Cloud-Based Tasking, Collection, Processing, Exploitation, and Dissemination in a Case-Based Reasoning System" proposes a novel computational intelligence methodology, which can learn to map distributed heterogeneous data to actionable meaning for dissemination. This approach provides a core solution to the tasking, collection, processing, exploitation, and dissemination problem. The expected performance improvements include the capture and reuse of analyst expertise, and, for the user, prioritized intelligence based on the knowledge derived from distributed heterogeneous sensing. "Simulation-Based Validation for Smart Grid Environments: Framework and Experimental Results" describes a simulation-based approach to understanding and examining the behavior of various components of a Smart Grid in the context of verification and validation. To achieve this goal, it adopts the discrete event system specification methodology, which allows the generalization and specialization of entities in the model and supports a customized simulation with specific scenarios.

Another issue is how to do sufficient formal specifications to support reliable construction and functioning of very large and complex systems. High-level representation mechanisms, including rigorous techniques for specification and verification, are needed. "An Institution for Alloy and Its Translation to Second-Order Logic" deals with the Alloy formal method for which it lays out the foundations to fully integrate the formalism in a platform which supports a huge network of logics, logic translators, and provers. This makes possible for Alloy specifications to borrow the power of several, nondedicated proof systems. "A Framework for Verification of SystemC Designs Using SystemC Waiting State Automata" presents the SystemC waiting-state automaton which is a compositional abstract

formal model for verifying properties of SystemC. It is first shown how to extract automata for SystemC components. Next, an approach is proposed to infer relations between predicates generated during symbolic execution. The correctness of the abstract analysis is proven by model checking. "Formal MDE-Based Tool Development" proposes a rigorous methodology to create formal tools for GUI-based domain-specific languages. It aims at providing a productive and trustworthy development methodology to safety critical industries. The methodology combines metamodel-based GUI generators with executable backends automatically generated from formal specifications. As for "Formal Modeling and Analysis of Learning-Based Routing in Mobile Wireless Sensor Networks," it presents a formal model for a learning-based routing protocol specific to wireless sensor networks. The model is based on a Bayesian learning method, using a Structural Operational Semantics style. It is analyzed by means of the rewriting logic tool Maude.

Reuse and integration are key concepts in information retrieval and data mining. They structure and configure the stored information in a way to facilitate its extraction and enhance its usefulness. "On the Use of Anaphora Resolution for Workflow Extraction" addresses the problem of workflow extraction from textual process descriptions and presents a framework to support the development of extraction applications. Resolution approaches are presented, and syntactic and semantic evaluation functions are developed. These functions which are based on precision, recall, and F-measure are used to assess the quality of the data-flow. Furthermore, the data mining community has turned a significant fraction of its attention to time series data. Virtually, the availability of plentiful labeled instances is assumed. However, this assumption is often unrealistic. Semi-supervised Learning seems like an ideal paradigm, because it can leverage the knowledge of both labeled and unlabeled instances. "A Minimum Description Length Technique for Semi-Supervised Time Series Classification," first, demonstrates that in many cases a small set of human annotated examples are sufficient to perform accurate classification. Second, it provides a novel parameter-free stopping criterion for semi-supervised learning. The experimental results suggest that the approach can typically construct accurate classifiers even if given only a single annotated instance.

Another key element in the success of reuse is the ability to predict variabilities. "Interpreting Random Forest Classification Models Using a Feature Contribution Method" presents an approach to show how feature contributions measure the influence of variables/features on the prediction outcome and provide explanations as to why a model makes a particular decision. It demonstrates how feature contributions can be applied to understand the dependence between instance characteristics and their predicted classification and to assess the reliability of the prediction.

In reuse, there is also a need for a seamless integration between the models output from domain analysis and the inputs needed to for domain implementations such as components, domain specific languages, and application generators. "Towards a High Level Language for Reuse and Integration" proposes a language

which gathers specialization and composition properties. The language is designed in a way to be specific to complex system domains. It supports, on the other hand, a component-based structure that conforms to a user-friendly component assembly. It is conceived in the spirit of SysML concepts and its programs generate Internal Block Diagrams.

Aspect orientation is a promising solution to software reuse. By localizing specific features in code aspects, not only a modular separation of concern is devised, but software development can also be incrementally transitioned to improve productivity and time-to-market. "An Exploratory Case Study on Exploiting Aspect Orientation in Mobile Game Porting" critically examines how aspect orientation is practiced in industrial-strength mobile game applications. The analysis takes into account technical artifacts, organizational structures, and their relationships. Altogether these complementary and synergistic viewpoints allow to formulate a set of hypotheses and to offer some concrete insights into developing information reuse and integration strategies.

Another area of potentially interesting research in reuse and integration is to identify what should be made reusable and which reusable corporate artifacts and processes will give the highest return on investment. "Developing Frameworks from Extended Feature Models" proposes an approach to develop a framework based on features defining its domain. The approach shows developers how to proceed, making them less prone to insert defects and bad smells in the outcome frameworks. It allows that even subjects with no experience in framework development can execute this task correctly and spending less time.

Researchers also argue for better methods to support specification and reasoning on knowledge component depositories. In "About Handling Non-conflicting Additional Information" the focus is on logic-based Artificial Intelligence systems that must accommodate some incoming symbolic knowledge that is not inconsistent with the initial beliefs but that however requires a form of belief change. First, the study investigates situations where the incoming piece of knowledge is both more informative and deductively follows from the preexisting beliefs. Likewise, it considers situations where the new piece of knowledge must replace or amend some previous beliefs, even when no logical inconsistency arises.

Safety and reliability are important issues which may be adequately addressed by reuse and integration. "A Multi-Layer Moving Target Defense Approach for Protecting Resource-Constrained Distributed Devices" proposes a Moving Target Defense approach for protecting resource-constrained mobile devices through fine-grained reconfiguration at different architectural layers. It introduces a coverage-based security metric to identify the configuration that best meets the current requirements. Likewise, in "Protocol Integration for Trust-Based Communication" a secure scheme based on trust is proposed to protect against packet dropping in mobile ad hoc networks. For this purpose, four already existing methods are integrated in a complementary way to the basic routing protocol in order to provide the required security.

Currently, most reuse research focuses on creating and integrating adaptable components at development or at compile time. However, with the emergence of ubiquitous computing, reuse technologies that can support adaptation and reconfiguration of architectures and components at runtime are in demand.

This edited book includes 15 high-quality research papers written by experts in information reuse and integration to cover the most recent advances in the field. These papers are extended versions of the best papers which were presented at IEEE International Conference on Information Reuse and Integration and IEEE International Workshop on Formal Methods Integration, held in San Francisco in August 2013. They have been selected among 111 accepted papers and have been accepted for publication in this book after they have been extended and have undergone a peer review process.

Thouraya Bouabana-Tebibel
Stuart H. Rubin

# Contents

**Cloud-Based Tasking, Collection, Processing, Exploitation, and Dissemination in a Case-Based Reasoning System** . . . . . 1
Stuart H. Rubin and Gordon K. Lee

**Simulation-Based Validation for Smart Grid Environments: Framework and Experimental Results** . . . . . 27
Wonkyu Han, Mike Mabey, Gail-Joon Ahn and Tae Sung Kim

**An Institution for Alloy and Its Translation to Second-Order Logic** . . . . . 45
Renato Neves, Alexandre Madeira, Manuel Martins and Luís Barbosa

**A Framework for Verification of SystemC Designs Using SystemC Waiting State Automata** . . . . . 77
Nesrine Harrath, Bruno Monsuez and Kamel Barkaoui

**Formal MDE-Based Tool Development** . . . . . 105
Robson Silva, Alexandre Mota and Rodrigo Rizzi Starr

**Formal Modeling and Analysis of Learning-Based Routing in Mobile Wireless Sensor Networks** . . . . . 127
Fatemeh Kazemeyni, Olaf Owe, Einar Broch Johnsen and Ilangko Balasingham

**On the Use of Anaphora Resolution for Workflow Extraction** . . . . . 151
Pol Schumacher, Mirjam Minor and Erik Schulte-Zurhausen

**A Minimum Description Length Technique for Semi-Supervised Time Series Classification** . . . . . 171
Nurjahan Begum, Bing Hu, Thanawin Rakthanmanon and Eamonn Keogh

**Interpreting Random Forest Classification Models Using a Feature Contribution Method** . . . . . 193
Anna Palczewska, Jan Palczewski, Richard Marchese Robinson and Daniel Neagu

**Towards a High Level Language for Reuse and Integration** . . . . . 219
Thouraya Bouabana-Tebibel, Stuart H. Rubin, Kadaouia Habib, Asmaa Chebba, Sofia Mellah and Lynda Allata

**An Exploratory Case Study on Exploiting Aspect Orientation in Mobile Game Porting** . . . . . 241
Tanmay Bhowmik, Vander Alves and Nan Niu

**Developing Frameworks from Extended Feature Models** . . . . . 263
Matheus Viana, Rosângela Penteado, Antônio do Prado and Rafael Durelli

**About Handling Non-conflicting Additional Information** . . . . . 285
Éric Grégoire

**A Multi-Layer Moving Target Defense Approach for Protecting Resource-Constrained Distributed Devices** . . . . . 299
Valentina Casola, Alessandra De Benedictis and Massimiliano Albanese

**Protocol Integration for Trust-Based Communication** . . . . . 325
Fatma Laidoui and Thouraya Bouabana-Tebibel

**Author Index** . . . . . 341

# Cloud-Based Tasking, Collection, Processing, Exploitation, and Dissemination in a Case-Based Reasoning System

**Stuart H. Rubin and Gordon K. Lee**

**Abstract** The current explosion in sensor data has brought us to a tipping point in the intelligence, surveillance, and reconnaissance technologies . This problem can be addressed through the insertion of novel artificial intelligence-based methodologies. The scope of the problem addressed in this chapter is to propose a novel computational intelligence methodology, which can learn to map distributed heterogeneous data to actionable meaning for dissemination. The impact of this approach is that it will provide a core solution to the tasking, collection, processing, exploitation, and dissemination (TCPED) problem. The expected operational performance improvements include the capture and reuse of analyst expertise, an order of magnitude reduction in required bandwidth, and, for the user, prioritized intelligence based on the knowledge derived from distributed heterogeneous sensing. A simple schema example is presented and an instantiation of it shows how to practically create feature search spaces. Given the availability of high-speed parallel processors, such an arrangement allows for the effective conceptualization of non-random causality.

**Keywords** Boolean features · Case-based reasoning (CBR) · Cloud-based tasking · Data exploitation · Schema instantiation

S. H. Rubin (✉)
SSC-PAC, San Diego, CA 92152-5001, USA
e-mail: stuart.rubin@navy.mil

G. K. Lee (✉)
Department of Electrical and Computer Engineering, San Diego State University,
San Diego, CA, USA
e-mail: glee@mail.sdsu.edu

T. Bouabana-Tebibel and S. H. Rubin (eds.), *Integration of Reusable Systems*,
Advances in Intelligent Systems and Computing 263, DOI: 10.1007/978-3-319-04717-1_1,
© Springer International Publishing Switzerland 2014

## 1 Introduction

The explosion in sensor data presents a massive challenge in the intelligence, surveillance, and reconnaissance technologies that may only be solved through the insertion of novel artificial intelligence-based methodologies. The nature of the problem is that heterogeneous data (i.e., structured and unstructured) must be reduced in size by an order of magnitude prior to dissemination. The focus of this position chapter is to suggest a novel computational intelligence methodology that can learn to map distributed heterogeneous data to actionable meaning for dissemination. This framework can provide a core solution to the tasking, collection, processing, exploitation, and dissemination (TCPED) problem. Further, it is expected that the approach will demonstrate the viability of the methodologies core concepts and thus justify a scaled-up investment in its development. The expected operational performance improvements include the capture and reuse of analyst expertise, an order of magnitude reduction in required bandwidth, and, for the user, prioritized intelligence based on the knowledge derived from distributed heterogeneous sensing.

To date, there have been many varied approaches to the TCPED problem ([1–7], for example). A common problem with these approaches is that either they employ *NP*-hard technologies that do not scale well (e.g., neural networks), or they attempt to apply logics, which are incomplete to tasks that inherently depend upon the development of quality heuristics—the learning of which is a science in itself.

Consider a schema-definition methodology; could such a strategy be scaled up for cloud-based computing? Will analysts who are non-programmers find it to be user friendly? Can the realized schemas for such applications as weather prediction or web searching find significant causal Boolean features? Boolean features are True/False responses to arbitrarily complex effective questions that collectively (along with at least one situational variable) define a situational context. Can schemas be symmetrically instantiated for greater speed of discovery? Can features be autonomously acquired, which enable correct predictions to be made that could not practically be made in their absence? While uncommon features may well be discovered (e.g., in the weather prediction application, finding changes in the temperature other than crossing the freezing boundary), we are looking to see that some common features are among them (e.g., changes in the barometric pressure).

According to Sam Fusaro [8], we are missing a capability for predictive analysis. The operational effectiveness of the approach presented here will bear proportion to the extent to which our approach can capture causality and thus model the cognitive process.

A fundamental principle of the tasking, collection, processing, exploitation, and dissemination (TCPED) is that intelligence processes must remain operational regardless of the amount of available bandwidth. A cloud-based system is needed to provide decentralized control. It is also advantageous because it is self-organizing and highly survivable. The goal of this chapter is to suggest that schema instantiation and case-based reasoning may be employed to resolve the large data processing issues in cloud computing. The approach presented here contributes to spectrum dominance

by evolving a feature-based understanding of the environment. One property of the proposed methodology is that it can be trained to meet most bandwidth constraints by virtue of its learning from skilled analysts. This is important for increasing the scale of the Intelligence, Surveillance, Reconnaissance, and Targeting (ISR&T) picture that will fuse data (and capabilities) from the cloud communities [9]. Furthermore, cloud computing will enable an individual client to utilize sensors regardless of their location.

Cloud computing has fewer problems than a network of heterogeneous machines. A grid computing system is suggested for the cloud computing system's back end. In this way, the cloud system can access the processing power of all available computers on the back end to make our methodologies potentially complex iterative calculations tractable. The cloud also integrates cloud communities that include the intelligence information collection communities.

## 2 An Illustrative Example

Suppose that we have the following case base, where $c_i$ are cases, $w_j$ are weights, the $i_j$ are situational variables (features), and d is the associated dependency class. Here, an asterisk, "*", represents a situational variable whose value is unknown, or was not recorded. Also, cases are acquired at the logical head, moved to the logical head when fired, and expunged from the logical tail when necessary to release space. Table 1 presents the schema for an arbitrary case base. The cases are shown in logical order, which is used by the uniform or 3-2-1 skew [10, 11]. The use of this skew is optional (i.e., in comparison with uniform weighting) and is useful for domains where the value of the data deteriorates in linear proportion to its time of collection—valuing more recent data more highly. The selection of a particular skew is domain specific. For example, the rate of radioactive decay is known to be proportional to how much radioactive material is left (excluding the presence of certain metals). The nuclear decay equation may be used as a skew for various radioactive materials and is given by $A(t) = A_0 e^{-\lambda t}$. Here, A(t) is the quantity of radioactive material at time t, and $A_0$ = A(0) is the initial quantity. The term λ is a positive number (i.e., the decay constant) defining the rate of decay for the particular radioactive material. A countably infinite number of other skews may be applicable.

In the following assignment of skew-weights, the skew vector, **S**, favors the logical head of the case base in keeping with Denning's principle of *temporal locality* [12]. Cases, which were most-recently acquired or fired, and thus appear at or nearer to the logical head of a case-base, are proportionately more heavily weighted under the 3-2-1 skew. Of course, this differs from a uniform skew. The closer a case is to the top of its linked list, the greater its weight or importance. A heuristic scheme (i.e., the 3-2-1 skew) for achieving this with a dependency class, d, consisting of $|d|$ cases is to assign the head case a weight of $\frac{2|d|}{|d|(|d|+1)}$. The map just below the head map has a weight of $\frac{2(|d|-1)}{|d|(|d|+1)}$. Finally, the tail map of the segmented case base

has a weight of $\frac{2}{|d|(|d|+1)}$. The $i$th map from the head has a weight of $\frac{2(|d|-i+1)}{|d|(|d|+1)}$, for $i = 1, 2, \ldots, |d|$. For example, using a vector of four weights, the 3-2-1 skew (**S**) is $\mathbf{S} = (0.4, 0.3, 0.2, 0.1)^{\mathrm{T}}$. There are a countably infinite number of possible skews, such that $\sum s_k = 1.0$.

The evaluation of the members of a dependency class is the contiguous weighted sum of its constituent elements (see below). A subsequent example will show how the weights are computed using the uniform and 3-2-1 skews, which again may be selected because they best fit domain characteristics. The weights are uniform if the skew is not known, or if there is no decay in the value of a case once recorded.

Table 1 shows a doubly-linked list. Zero indicates a list end. The list-head of the previous list is m and of the next list is one. The list-head of the free list (i.e., unused array elements) begins with the list-head of the previous list if the rows are fully utilized. Otherwise, the list-head of the free list points to the first row in the list of unutilized rows, in sequence. It simply contains every freed row, in arbitrary order.

Shift values are maintained for each non-Boolean variable (Table 1). These shifts are initialized to one minus the minimum field values, or zero—whichever is greater. If the resultant shift exceeds zero, each non-Boolean variable is initially shifted up by the computed shift value. Whenever a new contextual or situational variable has value less than or equal to the negation of its corresponding shift, then the shift takes the absolute value of that variable plus one. Non-Boolean variables not previously shifted (e.g., the context) will be shifted up by that amount, while all previously shifted ones (e.g., field values) will be shifted up by the values new—old shifts. Whenever a case is expunged, if the expunged non-Boolean variables have values of one, then new field minimums are found (i.e., an $O(\mathrm{m})$ process) and if their values exceed one, the associated shifts and the previously shifted variables are both reduced by the amount that those values exceed one. Thus, all non-Boolean non-asterisk variables will have value of at least one. This prevents divide-by-zero errors in normalization as well as problems in adjusting zero-valued weights.

Next, define a context by $c_j$ for j = 1, 2, …, n. The nearness of a pair of cases, $c_i$ and $c_j$, where the context is taken as $c_j$, is given by:

$$match(i) = \frac{\sum_{k=1}^{n} w_k \left| c_{i,k} - c_{j,k} \right|}{\left| participating\ situational\ variables \right|}, i \neq j.$$

It follows that since all weights and participating variable differences are normalized, $match(i) \in [0, 1]$. A participating situational variable is one that does not include an "*" in its field. If there are no such fields, then the pair of cases is omitted from the computation. If there are no such pairs of cases, then the match cannot be computed and thus is undefined.

The ranges of non-Boolean variables are normalized using double-precision computations. The non-Boolean vectors must be defined by positive elements. This is necessary to insure that any paired case differential, $|c_{i,k} - c_{j,k}|$, will never exceed unity. There must be at least one non-Boolean vector containing no asterisks. This is necessary to prevent divide-by-zero errors. The sums used for normalization are saved

**Table 1** The case base schema

| Wts | $w_1$ | $w_2$ | $w_3$ | … | $w_n$ | → |
|---|---|---|---|---|---|---|
| Ind | $i_1$ | $i_2$ | $i_3$ | … | $i_n$ | → |
| Feature | Non-B | Bool | Non-B | … | Bool | → |
| Shift | 0 | - | 0 | … | - | … |
| $c_1$ | 10 | 0 | 5 | … | 1 | → |
| $c_2$ | 15 | 1 | 10 | … | 0 | → |
| $c_3$ | * | * | 15 | … | 0 | → |
| … | … | … | … | … | … | … |
| $c_m$ | 5 | 1 | 10 | … | 1 | → |
| Wts | Dep. | Prv. | Nxt. | | | |
| Ind | D | - | - | | | |
| Feature | d | - | - | | | |
| Shift | - | - | - | | | |
| $c_1$ | 1 | 0 | 2 | | | |
| $c_2$ | 2 | 1 | 3 | | | |
| $c_3$ | 2 | 2 | m | | | |
| … | … | … | … | | | |
| $c_m$ | 1 | 3 | 0 | | | |

**Table 2** An arbitrary case base

| Wts | $w_1$ | $w_2$ | $w_3$ | $w_4$ | | Dep. |
|---|---|---|---|---|---|---|
| Ind | $i_1$ | $i_2$ | $i_3$ | $i_4$ | → | D |
| Feature | NB | Bool | NB | Bool | → | d |
| $c_1$ | 0.333 | 0 | 0.125 | 1 | → | 1 |
| $c_2$ | 0.5 | 1 | 0.25 | 0 | → | 2 |
| $c_3$ | * | * | 0.375 | 0 | → | 2 |
| $c_4$ | 0.167 | 1 | 0.25 | 1 | → | 1 |

for the subsequent normalization of any context. The sums for situational variables $i_1$ and $i_3$ in Table 1 are 30 and 40, respectively (asterisks are skipped). Normalization of these variables is shown in Table 2. Boolean and non-Boolean contextual differences will all be comparable (i.e., resulting in a uniform contribution of their importance, conditioned by their associated weight), since no paired case differential, $|c_{i,k} - c_{j,k}|$, will ever exceed unity.

The dependency class selected to be fired will be the weighted match (i), which has a minimal class value (see below). In the event of a tie, the dependency averaging (i.e., substituting the case dependencies relative position from the logical head for its match (i) value), nearer (at) the logical head of the case base is selected as the winner as a result of temporal locality [12]. The single case dependency, which is nearer (at) the logical head, is selected as the winner in the event of a second tie (e.g., d = 1 in Table 2 because $(1 + 4)/2 = (2 + 3)/2$, but $c_1$ is the logical head). Using 3-2-1 skew weighting, d = 1 wins again because $(2/3*1 + 1/3*4) < (2/3*2 + 1/3*3)$.

Relative fused possibilities are produced (e.g., using the uniform or 3-2-1 skew), which evidence that the decision to favor one class dependency over another may be more or less arbitrary. Of course, in some domains it may be more appropriate to present the user with an ordered list of alternative dependency classes, along with their weighted match (i) values, and let the user decide. The decision is necessarily a domain-specific one.

There are domains for which it is useful to know that the case base does not embody the desired matching case(s) and all but perhaps the slightest "guessing" is to be enjoined. This can be achieved by placing a squelch, greater than or equal to zero, on the minimum match (i). Here, if this minimum computed match (i) just exceeds the set squelch, then the system will respond with, "I'm very unsure of the correct action". The correct action dependency (d) will be paired with the context and acquired as a new case, at the earliest opportunity, if the dependency should prove to be incorrect. This dependency may or may not comprise a new action class. The logical tail (LRU'd member) of the case base may be expunged, as necessary, to make room for the new case acquisition. The qualifying phrases are, "very unsure", "somewhat unsure", "somewhat sure", "very sure", depending on the difference, (minimum match (d)—squelch, where d represents a dependency class). Notice that an exact match would have a difference of—squelch. Thus, any difference $<$ –squelch/2 would be associated with the qualifier, "very sure". Any –squelch/2 $\leq$ difference $\leq$ squelch/2 would be associated with the qualifier, "somewhat sure". Any squelch/2 $<$ difference $\leq$ squelch would be associated with the qualifier, "somewhat unsure". Finally, any squelch $<$ difference would be associated with the qualifier, "very unsure". The most appropriate value for the squelch may be determined experimentally and is domain specific.

Notice that the acquisition of a new case here not only insures that its situational context will be known until, if ever, the case falls off of the logical tail of the case base, but contexts in its immediate field (i.e., having minimal differences with it) will likewise be associatively known. Cases identified as erroneous may be (a) overwritten with the correct dependency, if known, (b) expunged, or (c) bypassed through the acquisition of a correct case at the logical head of the case base so that the erroneous case will eventually fall off of the logical tail of the case base. The choice of methodology here is domain specific in theory. In practice, alternative (a) is the usual favorite, where the domain is deterministic. Alternative (b) is the usual favorite, where correct actions are not forthcoming. Alternative (c) is the usual favorite, where the domain is non-deterministic (assumed herein).

Next, consider the arbitrary case base shown in Table 2. Here, we observe two dependency classes and two case instances mapping to each class. The goal is to find a normalized set of weights, $w_j$, which will serve in mapping an arbitrary context to the best-matching case situation and thus to the most appropriate action class, if any. We may take the squelch to be 0.1 here on the basis of trial and error for this example. If the minimum computed match (i) $>0.1$, then the system will reply to the effect that it is very unsure of the correct action. A correct case will be acquired at the logical head, if the correct action is known, should the dependency prove to be incorrect. In this event, the LRU'd case at the logical tail may be expunged to make room.

Ideally, each case situation is to be compared with each other case situation, exactly once, for the purpose of computing the global weight vector, **W**. The number of comparisons here is $m-1+m-2+\ldots+2+1$, or $\frac{m(m-1)}{2}$, which is $O(m^2)$ on a serial machine. This is only tractable for limited values of m. However, an excellent heuristic for reducing the order of magnitude of comparisons on large case bases without significantly reducing the quality of results is to compare the ciel (square root of the number of class members) with that defining number of members in each class including its own, though compared cases must be distinct. This is particularly useful for supporting the evolution of new features because it trades the time required for an exact evaluation of the weights for an (initial) rough estimation of any new features worth—allowing many more candidate features to be explored.

Once the feature set is stable, if ever, the full serial $O(m^2)$ class comparisons may be resumed. If m parallel processors can be utilized, then the runtime complexity here can be reduced to $O(m)$. This is certainly tractable, where we define tractability based on an efficient sort executing on a serial machine (e.g., MergeSort having average and worst case times of $O(N \log N)$ [13]).

The count of comparisons is made in order from the logical head, since these are the most recently acquired/fired cases and thus are intrinsically the most valuable based on temporal locality [12]. For example, in Table 2, cases $c_1$ and $c_4$ are members of the first class dependency and cases $c_2$ and $c_3$ are members of the second class dependency. Thus, $c_1$ is compared against $\lceil\sqrt{2}\rceil = 2$ members of its dependency class as well as two members of the second dependency class. Note that when a class member is being compared against its own class, the initial class member may be counted towards the computed limit in the number of comparisons (though skipped when i = j). Furthermore, if each class is dynamically linked based on order from the logical head so that one need not traverse other class members to get to the next member of the same class and each such class maintains a pointer to the next class in order, then, the number of comparisons is $\lceil\sqrt{m-1}\rceil + \lceil\sqrt{m-2}\rceil + \ldots + \lceil\sqrt{2}\rceil + 1$, which is $O(m)$ on a serial machine. (In practice, this is not too difficult to accomplish using two passes through the case base, or $O(m)$ extra storage.) If m parallel processors can be utilized, then the runtime complexity can be reduced to $O(\log m)$. If $m^2$ parallel processors can be utilized, then this can be reduced to $O(c)$, or constant time. Naturally, the use of $m^2$ parallel processors would only be practical for limited values of m.

Returning to our example in Table 2, $c_1$ will be compared against $c_2$, then against $c_3$, and next against $c_4$. Then, $c_2$ will be compared against $c_3$, then against $c_4$. Finally, $c_3$ will be compared against $c_4$. Note that $c_i$ is not compared against $c_i$ because this would serve to distort the resultant weights towards uniformity. Uniformity vies against variable (feature) selection. Also, the evolution of the weights is context free in terms of other weights. However, it is actually context sensitive because predicted dependency classes found to be in error are re-acquired as new cases having correct dependencies. The situational part of these new cases defines a new *well* for matching

similar cases [14]. Absolute values are not raised to any power on account of the availability of Boolean features, whose differences are immutable under any power. This also provides for faster computation as the power function is relatively slow in comparison with the arithmetic operators.

First, let us compare $c_1$ against $c_2$. These two situations belong to distinct classes (D). Here, we want to assign maximal weights, $w_j$, to the situational variables that are the most quantitatively different. Going across $w_1$ to $w_4$ in Table 2 for $c_1$ and $c_2$, $|c_1 - c_2|$ is computed as: $|0.333 - 0.5| = 0.167$, $|0 - 1| = 1$, $|0.125 - 0.25| = 0.125$, $|1-0| = 1$. We note that the qualitative features are at least as important as any other situational variable. This is evidenced by their more extreme absolute values relative to the non-Boolean situational variables. Here, $\mathbf{W} = (0.167, 1, 0.125, 1)$.

Next, let us compare $c_1$ against $c_3$. These two situations belong to distinct classes (D). Here, we want to assign maximal weights, $w_j$, to the situational variables that are the most quantitatively different. Going across $w_1$ to $w_4$ in Table 2 for $c_1$ and $c_3$, $|c_1 - c_3|$ is computed as: *, *, $|0.125 - 0.375| = 0.25$, $|1 - 0| = 1$. Here, $\mathbf{W} = (*, *, 0.25, 1)$.

Next, let us compare $c_1$ against $c_4$. These two situations belong to the same class (D). Here, we want to assign maximal weights, $W_j$, to the situational variables that are the most quantitatively the same. Going across $w_1$ to $w_4$ in Table 2 for $c_1$ and $c_4$, $|c_1 - c_4|$ is computed as: $|0.333 - 0.167| = 0.166$, $|0 - 1| = 1$, $|0.125 - 0.25| = 0.125$, $|1 - 1| = 0$. The Boolean situational variable differences need be complimented and the remaining non-asterisk variables subtracted from 1.0 because we need to weight the variables that are most similar most heavily. Here, $\mathbf{W} = (0.834, 0, 0.875, 1)$.

Next, let us compare $c_2$ against $c_3$. These two situations belong to the same class (D). Here, we want to assign maximal weights, $w_j$, to the situational variables that are the most quantitatively the same. Going across $w_1$ to $w_4$ in Table 2 for $c_2$ and $c_3$, $|c_2 - c_3|$ is computed as: *, *, $|0.25 - 0.375| = 0.125$, $|0 - 0| = 0$. The Boolean situational variable differences need be complimented and the remaining non-asterisk variables subtracted from 1.0 because we need to weight the variables that are most similar most heavily. Here, $\mathbf{W} = (*, *, 0.875, 1)$.

Next, let us compare $c_2$ against $c_4$. These two situations belong to distinct classes (D). Here, we want to assign maximal weights, $w_j$, to the situational variables that are the most quantitatively different. Going across $w_1$ to $w_4$ in Table 2 for $c_2$ and $c_4$, $|c_2 - c_4|$ is computed as: $|0.5 - 0.167| = 0.333$, $|1 - 1| = 0$, $|0.25 - 0.25| = 0$, $|0 - 1| = 1$. Here, $\mathbf{W} = (0.333, 0, 0, 1)$.

Next, let us compare $c_3$ against $c_4$. These two situations belong to distinct classes (D). Here, we want to assign maximal weights, $w_j$, to the situational variables that are the most quantitatively different. Going across $w_1$ to $w_4$ in Table 2 for $c_3$ and $c_4$, $|c_3 - c_4|$ is computed as: *, *, $|0.375 - 0.25| = 0.125$, $|0 - 1| = 1$. Here, $\mathbf{W} = (*, *, 0.125, 1)$.

Now, it is time to compute the normalized weight vectors. First, the raw computed weight vectors, $\mathbf{W}$, are given in Table 3. Next, each $c_{i,j}$ is normalized going across for $j = 1$ to n. Table 4 presents the results. The computed weights are then summed

**Table 3** The raw computed weights

| Wts | $w_1$ | $w_2$ | $w_3$ | $w_4$ | Init. $\sum$ |
|---|---|---|---|---|---|
| $c_{1,2}$ | 0.167 | 1 | 0.125 | 1 | 2.292 |
| $c_{1,3}$ | * | * | 0.25 | 1 | 1.25 |
| $c_{1,4}$ | 0.834 | 0 | 0.875 | 1 | 2.709 |
| $c_{2,3}$ | * | * | 0.875 | 1 | 1.875 |
| $c_{2,4}$ | 0.333 | 0 | 0 | 1 | 1.333 |
| $c_{3,4}$ | * | * | 0.125 | 1 | 1.125 |

**Table 4** The normalized rows

| Wts | $w_1$ | $w_2$ | $w_3$ | $w_4$ | Fin. $\sum$ |
|---|---|---|---|---|---|
| $c_{1,2}$ | 0.0729 | 0.4363 | 0.0545 | 0.4363 | 1.0 |
| $c_{1,3}$ | * | * | 0.2 | 0.8 | 1.0 |
| $c_{1,4}$ | 0.3079 | 0 | 0.323 | 0.3691 | 1.0 |
| $c_{2,3}$ | * | * | 0.4667 | 0.5333 | 1.0 |
| $c_{2,4}$ | 0.2498 | 0 | 0 | 0.7502 | 1.0 |
| $c_{3,4}$ | * | * | 0.1111 | 0.8889 | 1.0 |
| $\sum$ | 0.6306 | 0.4363 | 1.1553 | 3.7778 | 6.0 |
| Avg | 0.2102 | 0.1454 | 0.1926 | 0.6296 | 1.1778 |
| Norm | 0.1785 | 0.1234 | 0.1635 | 0.5346 | 1.0 |

and divided by the number of non-asterisked, situational variables and normalized to yield the final normalized four weights.

Suppose we had the situational context defined by $\mathbf{c}_1$ in Table 2. Clearly, the correct dependency class is 1. Let us see how this might be predicted.

First, let us compare $c_1$ against $c_1$. The raw context, $c_1 = (10, 0, 5, 1)$. Next, we perform a column-normalization by dividing the non-Boolean variables by the previously saved column sums. Thus, $c_1 = (10/30, 0, 5/40, 1) = (0.333, 0, 0.125, 1)$. Going across $w_1$ to $w_4$ in Table 2 for $c_1$, $|c_1 - c_1|$ is computed as: $|0.333 - 0.333| = 0$, $|0 - 0| = 0$, $|0.125 - 0.125| = 0$, $|1 - 1| = 0$. Thus, match (i = 1) = 0, which is minimal and thus is correctly matched. (Note that skewed averages are still necessary if non determinism is allowed. Here however, the case situation, $c_1$, occurs only once in Table 2—insuring that the correct dependency is d = 1).

Next, let us compare the situational context = (10, 1, 5, 0) against the four cases in Table 2 to find the best match and thus the predicted value for the dependency. Boolean matches are more significant and thus we would expect the dependency class to be 2 here.

Let us compare this situational context against $c_1$. Next, we perform a column-normalization by dividing the non-Boolean variables by the previously saved column sums. Thus, $c_1 = (10/30, 1, 5/40, 0) = (0.333, 1, 0.125, 0)$. Going across $w_1$ to $w_4$ in Table 2 for $c_1$, $|c_1 - \text{context}|$ is computed as: $|0.333 - 0.333| = 0$, $|0 - 1| = 1$, $|0.125 - 0.125| = 0$, $|1 - 0| = 1$.

$$\text{Thus, } match(1) = \frac{0.1785(0) + 0.1234(1) + 0.1635(0) + 0.5346(1)}{4} = 0.1645.$$

The denominator is four here because none of the situational variables union the context, in Table 2, has an asterisk and thus are all participating.

Next, let us compare this situational context against $c_2$. Going across $w_1$ to $w_4$ in Table 2 for $c_2$, $|c_2 - \text{context}|$ is computed as: $|0.5 - 0.333| = 0.167$, $|1 - 1| = 0$, $|0.25 - 0.125| = 0.125$, $|0 - 0| = 0$. Thus,

$$match(2) = \frac{0.1785(0.167) + 0.1234(0) + 0.1635(0.125) + 0.5346(0)}{4} = 0.0126.$$

Next, let us compare this situational context against $c_3$. Going across $w_1$ to $w_4$ in Table 2 for $c_3$, $|c_3 - \text{context}|$ is computed as: *, *, $|0.375 - 0.125| = 0.25$, $|0 - 0| = 0$.

$$match(3) = \frac{0.1785(0) + 0.1234(0) + 0.1635(0.25) + 0.5346(0)}{2} = 0.0204.$$

The denominator is two here because a total of two situational variables union the context, in Table 2, have asterisks and thus are not participating—being substituted for by zero above. Four situational variables reduced by two non-participating ones, leaves two participating situational variables.

Next, let us compare this situational context against $c_4$. Going across $w_1$ to $w_4$ in Table 2 for $c_4$, $|c_4 - \text{context}|$ is computed as: $|0.167 - 0.333| = 0.166$, $|1 - 1| = 0$, $|0.25 - 0.125| = 0.125$, $|1 - 0| = 1$. Thus,

$$match(4) = \frac{0.1785(0.166) + 0.1234(0) + 0.1635(0.125) + 0.5346(1)}{4} = 0.1462.$$

Next, we compute the uniform skew value for each class. $c_1$ and $c_4$ have d = 1 and $c_2$ and $c_3$ have d = 2. Thus, match ($class_1$) = (match(1) + match(4))/2 = (0.1645 + 0.1462)/2 = 0.1554. Match ($class_2$) = (match(2) + match(3))/2 = (0.0126 + 0.0204)/2 = 0.0165. Clearly, the second class (i.e., d = 2) is the better selection by a factor of almost ten. Here, the minimum match (i) – squelch = 0.0165 – 0.1 = –0.0835 < –squelch/2 and thus we would be "very sure" of it being the correct class. Also, the first class is above the set squelch of 0.1 and thus we would be "very unsure" of it being the correct class.

Had the 3-2-1 skew been used instead of the uniform skew, then the first member of each class would have been weighted more heavily, since it is closer to the logical head. The 3-2-1 skew only considers an element in positional relation to elements in the same class. The 3-2-1 skew for two elements is (2/3, 1/3). Thus, match ($class_1$) = $(0.6667 * 0.1645 + 0.3333 * 0.1462) = 0.158$. Match ($class_2$) = $(0.6667 * 0.0126 + 0.3333 * 0.0204) = 0.015$. Here, the use of the 3-2-1 skew has once again selected the second class (i.e., d = 2), as desired. We are slightly more sure of it being the correct class under the 3-2-1 than uniform skew because 0.015 – 0.1 = –0.085 < –0.0835 < –squelch/2.

**Table 5** The case base after acquisition

| Wts | $w_1$ | $w_2$ | $w_3$ | $w_4$ | | Dep. |
|---|---|---|---|---|---|---|
| Ind | $i_1$ | $i_2$ | $i_3$ | $i_4$ | → | D |
| Feature | NB | Bool | NB | Bool | → | d |
| $c_1$ | 0.333 | 1 | 0.125 | 0 | → | 3 |
| $c_2$ | 0.333 | 0 | 0.125 | 1 | → | 1 |
| $c_3$ | 0.5 | 1 | 0.25 | 0 | → | 2 |
| $c_4$ | * | * | 0.375 | 0 | → | 2 |
| $c_5$ | 0.167 | 1 | 0.25 | 1 | → | 1 |

Suppose however that it was learned that the correct dependency for the context was not d = 2, but rather something else—say d = 3, without loss of generality. Table 5 shows the resulting logical ordering of the updated case base. (Note that Table 5 would depict a nondeterministic case base if the context was an exact match for an existing situation, but the associated action differed.) If this table was limited to the storage of four cases, then case $c_5$ would be expunged to make room for case $c_1$. Physical case movement or search is not required. Again, this would involve updating the pointers to a doubly-linked global list. In Table 5, the global head pointer would be updated to point to case $c_1$ upon acquisition. Case acquisition or firing results in logical movement to create a new list head, where not redundant. The next pointer is updated to point to the previous head, or $c_2$. The next lower case having the same dependency, or case $c_5$, is eventually visited through global traversal. Counts of the number of members having the same dependency are maintained for the computation of the uniform or 3-2-1 skew. Case $c_5$ is last on the global list. Thus, it can be expunged by adding it to the free list and changing the last on the list to point to its immediate predecessor, or case $c_4$. Thus, the tail is updated to point to $c_4$.

Suppose that case $c_4$ were to be fired. It is logically moved to the head of the global list. Thus, the head pointer is set to $c_4$. Case $c_4$s next pointer is set to the previous head, or case $c_1$. Case $c_4$s predecessor's next pointer (case $c_3$) is set to point to case $c_4$s successor (case $c_5$).

Notice how the chance of generating an erroneous dependency (and its associated possibility) decreases with each case acquisition— given a segmented and relatively stable domain. This is because each case situation induces a proximal matching field for every context. The utility of this matching field is proportionate to the combined degree to which the situational variables were discriminators during training. This completes our example of case acquisition.

Next, consider the situational variables, $i_j$. Again, using m parallel processors, a runtime complexity of $O(\log m)$ is achievable. This is very fast and such speed can be put to good use in the evolution of new and better features. Table 4 shows the weights, $\mathbf{W}$ = (0.1785, 0.1234, 0.1635, 0.5346). In order of non-increasing significance they are $w_4$, $w_1$, $w_3$, and $w_2$. Observe that despite the missing data for the first two weights for one case (i.e., $c_3$ in Table 2), $w_1$ is not among the first two weighted-features to be replaced. $w_2$ is the least-significant non-zero weighted-feature, or the first to

be replaced. (Zero-valued weights need time to compute at least one comparative evaluation before being adjudicated.) We have seen that missing data does not affect the value assigned to a weight, as desired. The capability to handle this situation is generally required for feature evolution.

One evolutionary strategy is to run numerous distinct tables, having n situational variables each, in parallel. Each table holds predominantly distinct situational variables (features), though limited duplication is permitted to accelerate feature generation time. After all of the variables have received computed weights, the n greatest weights, and associated distinct variables (features) are concatenated in one table and renormalized. At least one non-Boolean situational variable containing no asterisks is required in the final table. Care must be exercised that no feature is "dangling"—due to reference to a missing situational variable(s), or even other feature(s). The requirement to compute new normalized elements requires that the original data (see Table 1) be the starting point each time—along with the appropriate sums (i.e., in view of a change in the included fields). This strategy is most useful where many parallel/distributed processors are available. More discussion on the use of parallel platforms and its need is provided in [15]. A stable sort, such as the n-way MergeSort [13], is used to sort the weight vectors. A stable sort maintains the relative order of records with equal values. This is important for feature evolution. Here, new situational features are inserted at the right. Thus, in the event of a tie, preexisting situational features will be more highly valued, since they have survived the longer test of time. Again, MergeSort has best and worst case times of $O$(N log N) and MergeSort's best case takes about half as many iterations as its worst case [13].

If no situational variable that is referenced by a feature(s) is allowed to be replaced and no feature that is referenced by another feature(s) is allowed to be replaced, then a second serial-processor strategy may be more efficient. The (non-zero lowest-weight) non-referenced variables and features can be replaced (or augmented) with new variables and/or features. To be replaced, the non-referenced variable or feature must have a weight, which is less than the average weight of all variables and features (except itself) having non-zero weights. Thus, better results can be expected if the number of features referencing other features is minimized (or eliminated). This is necessary to insure that relatively valuable non-referenced variables are not lost. Here, a few of the non-zero lowest-weight features can be found using a single-pass algorithm, which is O(n) on a sequential machine. The computation of Table 6 follows the same process as was illustrated for the computation of Table 3. The sums are unaffected. Tables 6 and 7 contain a replaced feature, $i_2$. The new feature, though far from perfect, evidences better discrimination than the one it replaced. The theoretical ideal weight vector, **W**, would have minimal cardinality such that all $w_j$ are equal. Ideally, a single situational variable would suffice to determine the dependency class. Table 7 computes the new weight vector, **W** = (0.1163, 0.2995, 0.1411, 0.4431).

Table 8 shows the acquisition of a new case by our arbitrary case base as set forth in Table 1. This case includes a new Boolean feature, $i_5$. Notice that this feature could not be computed for any but the new case due to the unavailability of data. Nevertheless, the system is designed so that the computed weight, $w_5 = 0$, can co-exist with the

**Table 6** The raw computed weights using the new w2

| Wts | $w_1$ | $w_2$ | $w_3$ | $w_4$ | Init. $\sum$ |
|---|---|---|---|---|---|
| $c_{1,2}$ | 0.167 | 1 | 0.125 | 1 | 2.292 |
| $c_{1,3}$ | * | * | 0.25 | 1 | 1.25 |
| $c_{1,4}$ | 0.834 | 1 | 0.875 | 1 | 3.709 |
| $c_{2,3}$ | * | * | 0.875 | 1 | 1.875 |
| $c_{2,4}$ | 0.333 | 1 | 0 | 1 | 2.333 |
| $c_{3,4}$ | * | * | 0.125 | 1 | 1.125 |

**Table 7** The normalized rows using the new w2

| Wts | $w_1$ | $w_2$ | $w_3$ | $w_4$ | Fin. $\sum$ |
|---|---|---|---|---|---|
| $c_{1,2}$ | 0.0729 | 0.4363 | 0.0545 | 0.4363 | 1.0 |
| $c_{1,3}$ | * | * | 0.2 | 0.8 | 1.0 |
| $c_{1,4}$ | 0.2249 | 0.2696 | 0.2359 | 0.2696 | 1.0 |
| $c_{2,3}$ | * | * | 0.4667 | 0.5333 | 1.0 |
| $c_{2,4}$ | 0.1428 | 0.4286 | 0 | 0.4286 | 1.0 |
| $c_{3,4}$ | * | * | 0.1111 | 0.8889 | 1.0 |
| $\sum$ | 0.4406 | 1.1345 | 1.0682 | 3.3567 | 6.0 |
| Avg | 0.1469 | 0.3782 | 0.1780 | 0.5595 | 1.2626 |
| Norm | 0.1163 | 0.2995 | 0.1411 | 0.4431 | 1.0 |

**Table 8** An acquisition for the arbitrary case base

| Wts | $w_1$ | $w_2$ | $w_3$ | $w_4$ | $w_5$ | | Dep. |
|---|---|---|---|---|---|---|---|
| Ind | $i_1$ | $i_2$ | $i_3$ | $i_4$ | $i_5$ | $\rightarrow$ | D |
| Feature | NB | Bool | NB | Bool | Bool | $\rightarrow$ | d |
| $c_1$ | 20 | 1 | 20 | 0 | 1 | $\rightarrow$ | 2 |
| $c_2$ | 10 | 0 | 5 | 1 | * | $\rightarrow$ | 1 |
| $c_3$ | 15 | 1 | 10 | 0 | * | $\rightarrow$ | 2 |
| $c_4$ | * | * | 15 | 0 | * | $\rightarrow$ | 2 |
| $c_5$ | 5 | 1 | 10 | 1 | * | $\rightarrow$ | 1 |

other weights, **W**, despite the single data point. In fact, having few data points (e.g., the square root of the number of class members), turns out to be an advantage for the evolution of new features. This is because having few data points supports a more uniform coverage of the feature search space. Most importantly, features are generated through the instantiation of schema, which involve situational variables and relational operators. Features may also be taken from case-base dependencies. A simple schema is presented in Fig. 1, discussed in more detail in [16].

```
Define Boolean Weather_Change_Feature (var x, t1, t2; t):
  /* In general, schemas may call other schemas. */
  Randomly Select x ∈ {pressure, humidity, temperature};
  Randomly Select t1, t2  ∈ {t, t-1, t-2, t-3}
    Such That t2 > t1;
  If x[t2] Randomly Select op ∈ {>, <} x[t1]
    Return (1)
Return (0).
```

**Fig. 1** A simple weather features schema

## 3 A Knowledge-Based Solution

The tasking, collection, processing, exploitation, and dissemination problem calls for a knowledge-based approach. The scope of this approach requires the acquisition and generalization of massive amounts of distributed knowledge to succeed. The nature of TCPED problems requires the piecing together of often diverse knowledge segments to arrive at a solution. The search for such solutions can be made tractable through the use of schemata (Figs. 1, 2, 3).

A simple schema is presented in Fig. 1. The search space for this schema is $3 \times 6 \times 2 = 36$ possible instances. The first random select statement allows for three choices (i.e., for pressure, humidity, or temperature). The six combinations for the second random select statement is derived from taking $n = 4$ things, $r = 2$ at a time, where the number of combinations, $c$, is defined by $c = \frac{n!}{r!(n-r)!}$. Finally, we see that there are two random choices for the final set of relational operators, $\{>, <\}$. Figures 2 and 3 show sample features, which are instances of their parent schema, which is found in Fig. 1. Again, they may be automatically discovered and validated through computational search. Figures 2 and 3 present one of 36 possible instances of this schema.

### *3.1 Case-Based Reasoning*

A case base consists of a set of situations and a sequence of actions such that the set is mapped to an appropriate sequence by way of experience, hence the term, experiential knowledge. This knowledge differs from rules in that it generally embeds causality, rather than literally explain it. Thus, cases are far easier to capture, maintain, and select for application.

Cases also differ from rules in that they are mapped to, rather than applied in the form of a logical inference engine. The problem is that it is generally impossible to directly capture causality. Any attempt to do so (e.g., through the use of rules) invariably leads to secondary interactions, which grow to become ever-more difficult to predict with scale. Cases are not associated with this difficulty because they are

```
Define Boolean Pressure_Increase_Feature (pressure, t):
  If pressure[t] > pressure[t-1]
    Return (1)
Return (0).
```

**Fig. 2** A "pressure" instance of the schema

```
Define Boolean Humidity_Decrease_Feature (humidity, t):
  If humidity[t-1] < humidity[t-3]
    Return (1)
Return (0).
```

**Fig. 3** A "humidity" instance of the schema

limited to the capture of experience, which may differ from the underpinning cause and effect. Case bases are also far less costly to maintain, for this reason, as has been borne out by industrial experience.

Automating TCPED processes requires the scalability and the ease of maintenance found in case-based systems, but needs the causal capture found in rule-based systems. The latter is necessary for the capture of analyst expertise and to minimize the number of case instances and thus the time needed for training the system.

## *3.2 An Example*

Situational knowledge consists of a set of conditional variables and Boolean features, which when satisfied imply a dependency. Dependencies define some system action and are indexed by class membership. For example, a UAV might run client-side software to record its GPS latitude and longitude (two variables), terrain data as reported by its sensors (variables); detect a road (feature), a river (feature), and an overpass (feature) [17]. This data (and potentially much more) is relayed to a cloud, which is running a weighted feature-based Case-Based Reasoning (CBR) system as a service. It maintains and evolves weights on all situational variables and features. If it finds no exact match for this mix of variable data and Boolean features, then in accordance with the novel methodology, it maps the data to the closest matching case.

This case implies a certain action, which is indexed and is a member of a specific (new) class. Moreover, a squelch is set to insure that the system will report when it does not know a proper match for a supplied context, or its level of confidence in the found match is below some preset threshold.

Here, that action class might be, "send text message to coalition cloud partner with UAV situational context + the tag, Boolean feature "Force Status?". The coalition cloud, running the same algorithm on a distinct case base, has evolved say high

weights on the GPS coordinate variables and the Force Status Boolean feature. This triggers its own member of an action class, say, "Friend". This is text messaged back to the cloud that the UAV is communicating with. This action, by definition, advertises a Boolean feature, "Friend", which augments (or substitutes into) the existing set of weighted situational variables and features. Now, the inference engine has the original set of six situational variables and features (five if one is replaced) plus this one. A new best-fit case is matched as before. (Note that the tagged Boolean feature, "Friend" need not be available and if not the firing pattern would be as previously described.) This time the triggered associated action class is say, "do not transmit imagery—all secure". Here, we see that operational bandwidth is conserved. The things to take away from this example is that if the following five conditions are all satisfied, then it will be practical to evolve dynamic solutions to the TCPED problems.

1. The cases can be rapidly acquired by the cloud servers.
2. Erroneous cases can be identified and expunged or updated.
3. The situational weights can be accurately evolved.
4. New features can be evolved, and/or received from fired dependency categories, to replace lower-quality features (i.e., those having the least non-zero weights).
5. The cloud servers can communicate using relatively low (available) bandwidth.

## *3.3 The Need for an Open Architecture*

The payoff from pursuing this approach is that problems can be solved that are not explicitly programmed for. The problem with expert systems is that the knowledge acquisition bottleneck [18] makes it impractical to avoid programming with scale (e.g., NASA's software tool for building expert systems—CLIPS). In the system proposed in this chapter, knowledge truly evolves from cases and the cases derive from analyst expertise. This expertise can be captured in the form of schema and exploited using cloud computing. The languages, used to define the schema, can be bootstrapped using this approach. The risk occurs if an open architecture is not utilized. This is because high-end schema-definition languages are more or less domain specific. Thus, they will improve over time and given an open architecture, the migration to upgrades, in the cloud, will be transparent to analysts on the front end. It is important that knowledge be retained through all upgrades.

## *3.4 Smart Tagging, Indexing, and Advertising*

Several institutions, including the US Navy, are pursuing cloud strategies for ISR/TCPED [19]. Processes are locally written and stored in the cloud. These processes can consist of Boolean features and effective dependencies. For example,

suppose a stream of data is sent to the cloud. We need smart and adaptive methodologies to manage mission data so we are not shipping the same product back five different ways and storing it 12 different times in accordance with current practice [20]. Rather, Boolean features are triggered by that stream. The matching set of situational variables and features triggers the best-matching case situation, which triggers the associated dependency if the certainty is above squelch (i.e., above a set minimum numeric quality criterion).

This cloud service does two basic things. First, it tags the data stream with a meaning (e.g., "Force Status?"). Second, it corresponds with a specific Boolean feature, which indicates that it was fired. That is, it advertises that the data was tagged with "Force Status?" indicated. Advertised data may be posted in any calling or called case base (i.e., intra and inter cloud). The risk here occurs where the memory space is too small to allow purchase of the advertised feature when even the least-weighted situational feature is needed and should not be arbitrarily lost. Another payoff of cloud computing is that adequate memory is generally not a problem.

This is potentially far more than "separating the wheat from the chaff" [19]. The approach to the problem in [19] is to process this data in situ so as to limit communication transmissions to mission critical data by autonomously extracting necessary information from the data stream amidst a sea of extraneous material. While this is helpful, we take the process to the next logical step. That is, we iteratively tag data streams and use the collective sequence of tags to direct further tagging. This is semantic scaffolding (also known as bootstrapping).

One rarely knows what data is needed so the iterative indexing of features effectively enables what is known in computer science as, "data-directed translation". Here, the various tags provide a context for subsequent tags. In human terms, this is building a cognitive picture. This feature, "Force Status?" always resets upon the next iteration of the inference engine. Tagging and advertising data are accomplished through the use of index tables, which are maintained in RAM for rapidity of access using limited communication paths. The resulting system rapidly evolves a correct complex behavior. This is attributed to the case interactions through smart tagging and indexing capabilities to advertise relevant data. The complexity of this process will be quadratic if each of m cases is compared against each other on a serial architecture, or linear in the cloud, where at least m distributed/parallel processors are available. However, if each case is compared against the square root of the number of cases in each class, then the complexity is reduced to linear on a serial architecture, or logarithmic in the cloud, where at least m distributed/parallel processors are available. This order of magnitude reduction in the number of ideal dependency category comparisons speeds up the discovery of new features by speeding up their evaluation and ranking. There will be insignificant impact on the quality of the resultant features because the cases, in each class taken for comparison, will be the most-recently acquired or fired (i.e., logically ordered). These cases will have the greatest utility.

## 4 On Boolean Features

Networks of case bases allow one bases' fired dependency category to serve as another's situational feature. Other features follow from the instantiation of domain-specific schema. For example, a temperature variable might need to distinguish the three natural states of water—solid, liquid, and gas. Here are a few tuples that serve to illustrate a simple schema and its instantiations, Schema: (Temperature°F, Freezing?, Boiling?), (32°, 1, 0), (72°, 0, 0), and (212°, 0, 1). The use of Boolean features is very common and effective as a descriptor.

As few as 20 Boolean features can provide universal situational identifications. For example, a parlor game exists called, "20 questions". Players take turns asking questions, which can be answered with a simple "Yes" or "No". The information, as measured by Shannon's entropy statistic, needed to identify an arbitrary object, from among 220 objects, is at most 20 bits [21]. The best questions will divide the field of remaining choices by two each time. Indeed, this is how binary search algorithms operate. Transforming this result, if binary decision points can differentiate an arbitrary object, then Boolean features can similarly characterize it. The better the questions, or equivalently the better the Boolean features the fewer that will be needed on average.

While these Boolean features are simplistic, they can be far more complex. For example, consider boosted/bagged neural networks, trained to recognize a particular face over streaming video. Note that if a network of architecturally distinct subsystems is trained, it is called, boosting. However, if identical subsystems are given distinct training, it is called, bagging. Schemas define alternative weight vectors, which program the neural networks recognition capabilities. Such weight vectors are obtained on the basis of external training. Furthermore, Zero Instruction Set Computer (ZISC) hardware can provide more than enough speed for real- time full-motion HD video Boolean feature recognition (e.g., less than 10 microseconds) [22]. Additional discussion on the use of Boolean features is given in [23].

## 5 The Role of Analysts

A new case is formed from the triggered features (i.e., both schemata instances and advertised dependencies) and the analyst-supplied action dependency. This machine training makes more efficient use of existing analysts through the capture and reuse of their expertise by the system as mediated through cloud-based architectures. The ISR tipping point continues to increase the demand for additional intelligence analysts to perform TCPED mission functions in traditional ways [20]. The expertise for the human piece of the intelligence analysis and exploitation that will be needed must be effectively captured for reuse, or we will not have the manpower to reduce the volumes of data that must be moved around.

Notice the advantage provided by iterative processing using an inference engine. That is, each iteration of the inference engine can transform the data stream by transforming it with the results of its processing (i.e., tagging). Such tagging can result in the iterative expansion and contraction of the grammatical object. This defines the universal, or Type 0, grammar. It then follows that no semantics is theoretically too sophisticated for it to effectively capture and process [24]. Intelligent systems that will not be found to be lacking in capabilities are needed in many applications.

Tags can also be formed by reaching back in the data stream and using previous processing results to tag the current stream. For example, once the data was tagged with "Force Status?" the set of features that is triggered on the next iteration of the inference engine recognizes that a request for information was sent to a federated cloud. What would a trained analyst do at this point? (S)he might notice from the reach-back that no hostile forces are in the vicinity. Knowing this and that no problems are indicated is sufficient to enable the analyst to tag the data stream with "Friend". This message is subsequently routed back to the cloud that made the request. This is defined as an executable tag, meaning that it is transmitted.

Notice that this methodology provides for the iterative fusion and reduction of multiple data streams using federated clouds running software as a service. Processing is done using the clouds platform as a service, which can execute in real time. One key to the success of the full- scale methodology is providing the analysts with an easy to use front end that (1) can capture their observations and decisions in software for reuse and (2) can execute as a service in the cloud environment. The creation of dependencies and most Boolean features does not require excessively complex software. Such software is said to be lightweight or thin-client because it can be composed exclusively on the client's platform.

## *5.1 Case Specification and Schema Definition*

The qualitative statement of intelligence needs derives transparently from the inference engine in finding the best match for the specified contextual situation. The user's intent and system objectives are embedded in a replay of analyst actions. They are not literally defined per se, but rather are an emergent property of system complexity.

Once schemas and cases are specified by the analysts, the evolution of better features is transparent. This is possible because the cases are partitioned into classes on the basis of their dependencies. Cases having the same dependency are assigned greater weights on their variables (features) that have near similar normalized values. Conversely, cases having distinct dependencies are assigned higher weights on their variables (features) that have the most different normalized values. This supports the evolution of the ideal feature set, occurs in the cloud, and converges on the capture of causality. Causality takes the form of a weighted set of appropriate Boolean features. Predicted dependency categories, found to be in error, are reacquired as new (nondeterministic) cases having correct dependencies. The situational part of these new cases defines a contextual well for matching similar cases [14].

The unique case-based weighted inference engine iteratively tags and removes tags from the data to evolve a simulation of our cognitive processes. Traditional CBR systems were designed to do this, but our novel CBR system autonomously evolves better features and more optimal weights. It also embodies data management to rank the utility of each acquired and fired case through logical movement. This also enables the square-root comparison method to work using relatively little data. Moreover, it can work with multiple alternative actions (i.e., non-determinism).

Knowledge acquisition is far easier than it would be using an expert system. Analysts could not work with the latter due to the inherent complexity of capturing causality. Moreover, unlike conventional programming, analysts need not be exact when specifying schemas. That is, the crispness associated with conventional computer programming is mollified to form a fuzzy space of alternatives. Our complex system may pass messages back and forth internally and between clouds before allowing a recommendation to emerge in satisfaction of the commander's intent and mission objectives. One property of the output of such a system is that it can be trained to meet most bandwidth constraints by virtue of its exercise by the analyst(s).

### *5.2 A Revolutionary Answer to an Evolutionary Need*

A recent TCPED study estimated a future need for over 300 additional intelligence analysts [20]. Their knowledge and skills need to be captured for replay in training the cloud-based CBR systems. Otherwise, adding more analysts is an evolutionary answer to a revolutionary need. Analysts may opt to use sophisticated, but easy to work with agent based modeling toolkits to parse the data and insert the proper tags (e.g., ABLE, SOAR 6 in JAVA, SimPlusPlus in C++, but no programming required, et al.). They will also write top-down and/or bottom-up schema (see below), which are used to constrain the feature space. A treatise on the use of experts and analysts is given in [15].

## 6 On Unsupervised Feature Learning

While schema definition effectively provides for semi-supervised learning because the machine is dependent on the human and vice versa, the question arises if the unsupervised learning of features is practical. First, in order to be tractable in the large, knowledge is required to guide search. Thus, our question at once is transformed into, "What is the most efficient representation(s) for knowledge?" and, "What is the paradigm(s) for knowledge acquisition?"

The answer to the first question is to employ more constrained schema, more parallel computation, and a better way to associate the problem definition with an associated schema. Top-down and bottom-up methods have been given for schema generalization. These methods depend upon having a symbolic representation.

This representation needs to be composable to create a search space of alternatives. The selection among alternatives is knowledge based (e.g., using rules or preferably cases).

The answer to the second question is to use the same case-based weighted feature acquisition system. In other words, the dependencies at this level may be feature schema, which are more general than the independent features leading to their selection. These dependencies capture greater knowledge than embodied by the independent features leading to their selection.

Next, we consider the basis for feature-based schema selection. Features are defined by the absence of noise [14, 17–30]. In other words, features are randomizable. Such randomization can be lossless (i.e., invertible), or with greater or lesser degrees of loss (e.g., in the "real" world). For example, an upper right angle and a lower right angle can be learned by a neural network and associated with the concept of right angle. However, the learned information is not compressed and in this sense it is not randomized. Hence, we say that a feature has not been extracted. All manner of methods have been used to extract the concept (e.g., positive and negative examples, grammatical inference, etc.). However, they all fall short in what we refer to as conceptualization.

Patterns may be stored and recursively adapted within limits to match meta-patterns. New patterns may be learned. In essence, such mapping procedures can facilitate recognition. But, randomization is not limited to structural randomization. It may be functional as well. For example, a large flat rock and a chair are functionally randomizable. Again, we see the need for symbolic representations of knowledge. While symmetric knowledge can be discovered through the use of an inference engine (e.g., a sandy beach is like a bed), random knowledge cannot. It must be limited to the product of exhaustive search. Such search must involve conceptual tokens (i.e., words and phrases) because only they are randomizations of the intended concept (e.g., neural weights are not randomizations). The compression of fundamental memories in a neural network requires context (e.g., is it that they are right angles or that they are sharp?). As such, context must be separable. This can only be accomplished through the use of symbolic representations. This suggests the use of neural networks as a frontend for symbolic representation and the symbolic composition of schemata using system dependencies (i.e., knowledge-based composition of schemata).

Sometimes a single schema is sufficient for an instance of it to capture a general concept. At other times, a more or less linear independent association of schema instances is necessary to capture a general concept. This is because randomization is an iterative process. The end point of randomization is recursively enumerable, but not recursive. Thus, while fewer schemata instances are necessary over time, one never truly knows if a relative minimum has been attained. That is why conceptualization is time dependent [23].

## 7 Five Research Challenges

There are five research challenges, amongst others, which need to be addressed [16].

1. Develop a schema definition methodology suitable for small-scale testing and realizable software to facilitate the implementation of self-modifying code.
2. Investigate and report on how to best constrain the implied search.
3. Investigate tagging using advertised features (i.e., self-referential feedback in machine learning)—this will also prove the utility of intra and inter cloud communication if successful.
4. Evaluate the quality of the evolved features as determined by the evolution of their associated weights towards a set having minimal statistical variance (i.e., a surrogate criterion for high utility features). Also, if the number of descriptive features is allowed to vary then a successful feature evolution will usually, but not always reduce the number of maintained features. (While having a minimal number of maximally-descriptive features is desired, that minimum is domain specific and cannot be known in the general case.)
5. Evaluate the subjective quality of the feature evolution process—including the number and relevance of the features evolved.

With research challenges, one must also ask some questions which may lead to potential reinforcements of proposed approaches or lead to other approaches not initially considered.

1. Can the schema-definition methodology utilize the massively distributed environment provided by cloud computing? Can libraries of schema templates be defined and used as is common for object-oriented programming languages? Will schemas be easy for analysts to use to constrain the search space? Can random instances of analyst-specified weather prediction schemas and/or symmetric instances of analyst-specified weather prediction features yield significant causal Boolean features?
2. That random instantiation of schemas works was empirically demonstrated in 1998 in an unpublished computer program. Will the random instantiation of schemas, or the symmetric extension of features, if either, prove to be a more productive methodology for analyst use? This topic is suitable for a patent disclosure(s) and/or research paper(s). It is likely to be best realized through economies of scale. Can Bernstein's concept of multiple analogies, where having multiple derivational paths increases the likelihood of having a valid result, be empirically confirmed.
3. Can correct (i.e., meaningful) weather predictions be made using two or more cycles of the inference engine such that at least one advertised Boolean feature tag, which was not present in the first cycle, and is subsequently posted, leads to the firing of a correct action dependency? This demonstrable event, which implies successful schema definition and instantiation, provides clear indication that the methodology can scale, be cloud-based, and address TCPED intelligence processes. That is the impact. The needed bandwidth will be relatively low

because text, not say HD video, will be streamed, for the most part, over the network.

4. Ideally, a minimum number of features will all have about the same weight and suffice to determine the proper dependencies. Statistical variance, in the weights, will be computed along a path from the initial weight vector and associated feature set to an evolved weight vector and associated feature set. A plotted approximation of a decreasing exponential curve will evidence that the quality of the Boolean features is successfully improving. Furthermore, if the number of maintained features is allowed to vary based on the magnitude of the associated evolved weights, then a decreasing number of maintained features, in general, will likewise support the improving quality of the evolving features.
5. The features evolved for the application domain (e.g., weather prediction) will be evaluated by their number (less is better), their relevance as defined by their associated weights (greater is better), and most importantly by their use in the actual domain. While uncommon meteorological features may well be discovered here (e.g., changes in the relative humidity), we are looking to see that some common meteorological features are among them (e.g., changes in the barometric pressure). Freely available weather history reports can be obtained from the Web, from <http://weathersource.com/past-weather/weather-history-reports/free>. This data can be used to support feature evolution. Successful evaluation of the five categorical questions collectively provides clear evidence that the proposed methodology works as described. Each loosely coupled CBR system in the cloud will benefit from each other's quality improvements. The features and their weights are continuously evolved on the back end (e.g., the cloud's grid computing system) to dynamically optimize performance.

The approach suggested here advances beyond others research by providing the following ten fundamental capabilities.

1. It evolves causal behavior—it does not require the user to find rules to specify it.
2. It utilizes schema-definition languages to define and evolve candidate features.
3. Candidate features are also automatically derived from every fired dependency. This is also supported by low bandwidth intra and inter-cloud networking.
4. It compares cases against the same as well as distinct classes of case dependencies to evolve near optimal features and associated weight sets for the supplied TCPED problems.
5. The weight sets may be rapidly evolved, using a novel "square-root class comparison method" to more quickly converge upon the best features.
6. Cases are acquired, fired, and expunged to free space using dynamic memory management and logical (i.e., not physical) movement for speed.
7. System training is transparently provided by teams of distributed analysts, using intelligent agents so that their expertise is readily captured for replay.
8. Every action is associated with an evaluation of its certainty. Actions having low certainties are subject to being squelched (i.e., filtered).

9. It learns from analyst training to iteratively find meaning in multiple data streams. Dependency actions can disseminate this meaning, in the cloud, using an order of magnitude less bandwidth. This is made possible through the use of smart tagging and advertising.
10. All of the novel advancements above utilize the massive parallelism available through platform as a service using federated cloud architectures.

A successful approach for conceptualizing causality, and thus modeling the cognitive process, will define a solution for the entire intelligence process, or TCPED. There is a small risk of failure here. However, the methodology overviewed herein shows that TCPED intelligence processes can be automated to a much greater extent than is presently possible. That is the payoff.

## 8 Conclusion

Determining causal relationships from observations and experiments is fundamental to human reasoning, decision making, and the advancement of science. In the final analysis, this methodology converges on effectively capturing causality—to the extent that it is not randomly based. It shows that the evolution of features and the machine learning of cases provide a unified framework for the capture of intelligence. Furthermore, the definition of schemas, for the production of novel features, is defined by randomness and symmetry [11, 25, 27].

The methodology is robust and can handle limited noise and/or missing data. In particular, evolution can occur on top of an existing case base—even if the data needed to evaluate the new variables (features) cannot be had for the preexisting cases. A squelch is set to insure that the system will report when it does not know a proper match for a supplied context. Cases are logically acquired at the head of the base and moved there whenever fired. The least-recently used (LRU'd) cases are expunged from the tail when necessary to free space

Weights are evolved by processing each case against each other in a segmented case base and partitioning the cases on the basis of having the same or distinct class dependencies. Greater weights are used if the situational variables being compared are different and the associated action dependencies belong to distinct classes. Similarly, greater weights are used if the situational variables being compared are similar and the associated action dependencies belong to the same class. Situational variables are normalized and unified with Boolean features. The methodology can be O(log m) in runtime complexity if m parallel processors are utilized and the square root of the number of class members is used for comparison purposes. This allows for the rapid evolution of new features—decreasing the entropy of the supplied data. Contexts are mapped to the class whose components have minimal weighted error (i.e., evaluate closest to zero). These components are typically uniformly weighted, or weighted using the 3-2-1 skew. The latter reflects the age-weighted properties of