Kenneth J. Turner, Alex M. Gillespie and Lynne J. McMichael.
Rigorous Development of Prompting Dialogues, Biomedical Informatics,
44(4):713-727, September 2011

Rigorous Development of Prompting Dialogues

Kenneth J. Turner[a], Alex Gillespie[b], Lynne J. McMichael[a]

[a] *Computing Science and Mathematics, University of Stirling, Stirling FK9 4LA, UK*
[b] *Psychology, University of Stirling, Stirling FK9 4LA, UK*

Abstract

Objectives The aim was to support people with cognitive impairment through speech-based dialogues that guide them through everyday tasks such as activities of daily living. The research objectives were to simplify the design of prompting dialogues, to automate the checking of prompting dialogues for syntactic and semantic errors, and to automate the translation of dialogue designs into a form that allows their ready deployment.

Approach Prompting dialogues are described using CRESS (Communication Representation Employing Systematic Specification). This is a notation and toolset that allows the flow in a service (such as a dialogue) to be defined in an understandable and graphical way. A dialogue diagram is automatically translated into a formal specification for rigorous verification and validation. Once confidence has been built in the dialogue design, the dialogue diagram is automatically translated into VoiceXML and deployed on a voice platform.

Results All key objectives of the work have been achieved. A variety of significant dialogues have been successfully represented using the CRESS notation. These dialogues have been been automatically analysed through formal verification and validation in order to detect anomalies. Finally, the dialogues have been automatically realised on a VoiceXML platform and have been evaluated with volunteer users.

Keywords:
Cognitive Impairment, CRESS (Communication Representation Employing Systematic Specification), Dialogue, Formal Method, GUIDE (General User Interface for Disorders of Execution), IVR (Interactive Voice Response), LOTOS (Language Of Temporal Ordering Specification), Prompting, Validation, Verification, VoiceXML

1. Introduction

This section discusses the motivation and objectives of the research. As healthcare background, the nature and implications of cognitive impairment are explained. As technical background, the design of dialogues, prompting systems and formal methods for these are discussed.

Email addresses: kjt@cs.stir.ac.uk (Kenneth J. Turner), alex.gillespie@stir.ac.uk (Alex Gillespie), lynnemcmichael191@btinternet.com (Lynne J. McMichael)

1.1. Motivation

Cognitive impairment is widespread and has various causes including dementia, stroke, traumatic injury, learning difficulties and mental illness. People with cognitive impairment can have problems initiating, planning, sequencing, attending and remembering. Severe cognitive impairment can make it impossible without support to perform routine activities of daily living such as food preparation, dressing, laundering and self-care.

Technological support for people with cognitive impairment is therefore highly desirable. Research has shown that appropriate technology can support cognitive function and thus enable independence [28]. Assistive technology for cognition has been shown to help people with dementia, traumatic brain injury and cerebrovascular accident. Several reminding devices have been developed to help people with daily tasks.

More recently, researchers have begun working on micro-prompting (or sequencing) systems that guide users step-by-step through a given activity [19, 34, 36]. It has been argued that the optimum strategy is to simulate the verbal prompting provided by carers [36]. This is because verbal prompts do not interfere with visual tasks, do not increase cognitive load, and do not require mastery of any new technology by the user [35]. The theory and rationale behind prompting dialogues is discussed in section 1.3.1.

Prompting for even relatively ordinary tasks requires careful design of complex dialogues. These are typically prepared by care professionals or family members. Errors in prompting dialogues are undesirable as they are likely to confuse a user who is already struggling to complete a task. Incorrect dialogues can also raise safety concerns (e.g. an incorrectly donned artificial limb might cause a fall or inflammation). There is, as yet, no established procedure for ensuring that the dialogues used in prompting devices are free from error.

The starting point for the work in this paper was the GUIDE prompting system (General User Interface for Disorders of Execution, described in Section 1.3.3). This provided a set of dialogues for study, and a baseline for comparison with the new approach in this paper. The new work aimed to improve on GUIDE as follows:

- to simplify the design of prompting dialogues
- to automate the checking of dialogues for technical errors in syntax and semantics
- to automate the translation of dialogue designs into a form that allows their ready deployment.

The methodology of this paper is generic in two senses. Firstly, it applies to rigorous design of many kinds of dialogues and to interactive voice response systems in general. Secondly, it uses CRESS (Communication Representation Employing Systematic Specification, Section 2) as a general approach to rigorous design of many kinds of services (dialogues being only one kind of example).

All the key techniques and tools have been developed by the authors: the overall methodology, the dialogue design principles, and the CRESS toolset for creating, analysing and realising dialogues. However, these rely on other general-purpose tools developed by others: analysis tools for the LOTOS specification language, and speech tools for the VoiceXML scripting language.

1.2. Healthcare Background

The cost of formal and informal care provision per annum is over US$300 billion in the USA [26] and over £66 billion in the UK [13]; these costs are becoming unsustainable [5]. The majority of this care is for basic activities of daily living such as dressing, personal hygiene and food preparation. In supporting cognitive impairment, care providers mainly monitor activity performance and provide verbal prompts.

The social and identity cost of care is also significant. Cognitive impairment can be embarrassing and distressing, often leading to feelings of invasion of privacy, dependency and being treated like a child [41]. It is important for the individual to retain a sense of identity and feelings of self-worth.

When supporting people with cognitive impairment, carers have been observed to follow a wide variety of prompting strategies. Sometimes carers provide only verbal prompts, sometimes they model the desired activity, and sometimes they use 'hand-over-hand' support [21, 44]. A study was made of verbal prompts provided by formal caregivers to people with Alzheimer's disease during a hand washing task [64]. This found single-proposition prompts to account for almost half of all prompts, and that closed questions, repetition and paraphrased repetition were also common.

Carers provide verbal prompts that remind the care receiver of what to do and how to do it. The care receiver can experience this verbal support as overprotective, nagging or undermining [20]. Caring for someone with a cognitive impairment can be a tremendous strain for the caregiver, leading to stress, depression, anxiety, lack of sleep and fatigue [41].

Although cognitive impairment can affect all ages, it particularly affects older people (who are most prone to dementia and stroke). In the UK, the percentage of the population aged 65 or over is expected to increase from 16% in 2009 to 23% in 2034 [59]. However the most significant increase will be in the age group of 85 years and over (5% of the population by 2034).

Dementia is the largest cause of cognitive impairment. It describes a group of symptoms associated with a progressive decline in memory, understanding, judgement, language and thinking. In the UK, the number of people with dementia is currently over 821,000 (1.3% of the population) [29], and predicted to be 1,740,000 by 2051. According to [65], the global cost of dementia in 2010 was US$604 billion. An 85% increase in this is predicted by 2030, with the bulk of the cost relating to care provision.

Medical interventions aimed at restoring cognitive disabilities in dementia have had limited success [15]. Biomedical attempts to find a 'cure' for dementia are also problematic [30]. At best these interventions delay the onset of symptoms, possibly prolonging the period of dependency on care. A solution is needed to the problem of care, not to the problem of nerve degeneration.

Brain injury is another cause of cognitive impairment. Around 500,000 people in the UK live with a long-term disability as a result of a traumatic brain injury [22]. Over 143,000 people in the UK have a stroke each year, the majority being over 65. 75% of stroke survivors experience disability in physical, emotional or cognitive functions.

Learning disabilities make it difficult to learn as quickly or in the same way as an unaffected individual. The number with learning disabilities in the UK is currently estimated to be 1,105,000 [17]. Again, the ageing population means that a 36% increase is expected from 2001 to 2021.

The above statistics reveal the massive scale of cognitive impairment, with its resultant economic and social costs. Given the limited potential for biomedical cures, technology is increasingly seen as a means of transforming the provision of care. A central aspect of such technologies will likely be prompting dialogues aimed at emulating the cognitive support already being provided by caregivers.

1.3. Technical Background

1.3.1. Prompting Dialogues for People with Cognitive Impairment

People routinely provide cognitive support. For example, parents monitor the activity of their children and intervene with suitable verbal suggestions [11]. The process through which an expert guides a novice in a task, using primarily verbal support, has been called 'scaffolding' [38]; this has been extensively studied in developmental psychology [68]. Verbal scaffolding entails the expert reminding the novice, focusing attention, and helping to conceptualise and sequence a task.

Following the principles of scaffolding, prompting dialogues are defined to involve an expert providing a verbal 'scaffold' just beyond the ability of the novice. This allows novices to perform above their own unaided ability. The difference between unaided and aided ability was termed 'the zone of proximal development' by Vygotsky. It has been argued that encouraging action within this zone is essential to development [60]. Exactly how verbal prompting interacts with cognitive function is unclear. A Vygotskian standpoint assumes that higher mental functions are largely verbally mediated through truncated internal dialogues. It is then possible that verbal prompting directly supports cognitive function. For example, verbal prompting within the zone of proximal development often uses questions. It might be that these stimulate self-reflection in the novice, scaffolding self-questioning and thus self-regulation of behaviour.

Recently the concept of scaffolding has been used to understand the verbal support provided by carers and therapists to people with cognitive impairment during task performance [35, 38]. Therapists and carers working with people who have sequence performance difficulties can be conceptualised as providing external support for initiation, problem-solving, generativity, planning, sequencing, organisation, self-monitoring, error correction and behavioural inhibition. To benefit from this instruction, patients require different and often intact cognitive processes such as verbal comprehension, object identification, memory of single stage directions, and verbally mediated motor control [67].

The concepts of scaffolding cannot be transposed without modification from supporting cognitive development in children to supporting cognitive function in adults [46]. Moreover, the prompting strategies that caregivers report they follow do not agree with the actual strategies they are observed to use [44]. The following design principles for prompting dialogues are based on a review of the literature about providing verbal prompts to adults with cognitive impairment:

- the prompts should be in line with expectations and unambiguous [16]
- prompts should use common sense and everyday landmarks [7]
- key words should be preceded by a pause or a disfluency such as 'er' [12]
- the use of metaphor should be avoided [66]

- prompts should be phrased in terms of 'do' rather than 'don't' [1]
- speaking slowly is not necessarily beneficial, because it makes prompts longer and thus more taxing on memory [44]
- prompts should use as few prepositions as possible [62]
- prompts should begin with the user's name in order to get attention [34]
- prompts should use a male voice as high frequencies may be harder to hear [33]
- prompts should be repeated regularly because sentence recall is very poor, although sentence comprehension can be good amongst people with dementia [8]
- key prompts should begin with alerting redundancy (e.g. 'That's great, now do ...') to give a user sufficient time to attend to the incoming information [36]
- yes/no questions are more effective at preventing communication breakdown than open-ended questions, but they can undermine the freedom of the person with cognitive impairment [44].

It is not expected that all these prompting strategies will be appropriate for all types of cognitive impairment, for all levels of severity and for all tasks. As cognitive impairment becomes more severe, research has shown that prompting becomes more verbally directive (more commands and fewer questions), and there is an increase in visual and even physical prompting (e.g. hand-over-hand type prompting) [44]. However, even people with relatively severe dementia can benefit from verbal prompting [2, 42]. When a task is unfamiliar then physical modelling can be helpful, while a familiar task that requires visual attention may benefit most from verbal prompting [35]. As an example, a navigation task was set for people with severe acquired brain injury [45]. Audio direction without a map proved more effective than use of an aerial view map, a point-of-view map, and textual instructions.

The CRESS approach in this paper for designing prompting dialogues is not dependent upon any particular prompting principles. It can be used to implement a wide range of prompting strategies as indicated by the context.

1.3.2. Prompting Systems

Assistive technology for cognition, in particular prompting, is not a new concept. There have been numerous research studies into this subject, and several prompting systems already exist. [9] provides a review of memory aid devices for older users.

Schedulers are designed to remind someone with cognitive impairment of tasks to be performed (e.g. attend an appointment or take medication). Examples include NeuroPage [63], MemoJog [23] and MEMEX [40]. These devices allow users to define schedules for tasks such as preparing for a visit from the therapist or cooking a meal. Text prompts are then sent to the individual via a PDA (Personal Digital Assistant) or mobile phone when a task is due. These devices have been shown to increase the ability of the individual to achieve target behaviours. Although the devices are useful for reminding an individual to perform a task, they do not provide step-by-step support in how to perform the task.

Sequencing systems can be used to provide step-by-step support and guidance to carry out daily tasks. People with a cognitive impairment can find it difficult to plan

and sequence the key actions in an activity of daily living. Sequencing devices aim to assist the individual's memory by placing task steps in an appropriate order. In effect they prompt the individual through the steps required. One of the earliest solutions was PEAT (Planning and Execution Assistant and Trainer [27]). This provides the user with daily plans by making use of artificial intelligence. The system cues the user when to start or stop a task, monitors and records task performance, has a mechanism to adapt to schedule changes, and has task scripts to guide the user through some activities of daily living. More recently, researchers have begun working on micro-prompting (or sequencing) systems that guide users step-by-step through some activity [19, 34, 36].

Essential Steps [10] is a software package that uses on-screen cues and a computer generated voice to guide the individual through various tasks. MAPS [10] and the commercial Pocket Coach [19] allow use of a desktop computer to create mainly visual prompts. These are then stored on a PDA that prompts the user through the activities. The user can respond to prompts by pressing buttons on the PDA. Although these devices do show an improvement in target behaviour, they require the individual to first learn the system before they can use it. Furthermore, the individual often has to interact with a complex and unfamiliar interface. It has been argued that assistive technology devices can increase cognitive burden, not reduce it [28].

The extent to which assistive technology can aid people with cognitive impairment depends very much on how willing the individual is to use the device. This in turn depends on how useful the individual or the carer finds the device, how easy it is to use, and whether or not the device supports a sense of personal identity [32]. To be useful to both individuals and their carers, assistive technology must be autonomous, non-invasive, and not require explicit feedback such as pressing buttons.

COACH (Cognitive Orthosis for assisting Activities in the Home [34]) was developed in response to this need. The aim was to create a device that uses minimal hardware and does not require any input from the user. The system uses artificial intelligence to independently guide the user through the activity of hand washing using audio and video prompts. COACH was evaluated by six older people with moderate to severe dementia. The results showed that 11% more hand washing steps were completed independently, and that there were 60% fewer carer interactions. Although the results showed promise, it was concluded that the number of participants in the study was not large enough to draw any significant conclusions regarding widespread applicability.

A common factor in all these approaches is a heavy reliance on visual cues for prompting. This requires users to divert their attention away from the task they are performing to look at prompts or cues on a visual display. It can be difficult if the individual has to constantly look at a screen for prompts when they are not free to do so, e.g. while cleaning the house or dressing. Recent research has concluded that prompts should be more in line with how a carer might prompt an individual, i.e. verbally and not visually [36]. The verbal support that a carer provides to an individual with a cognitive impairment is familiar and natural. Therefore any approach mimicking this support should require almost no familiarisation. COACH initially provides an audio prompt (similar to a carer) and then an audio-visual prompt. However there is no way for users to interact with the system, e.g. they cannot indicate whether they have completed a task successfully or not.

1.3.3. The GUIDE Approach

GUIDE (General User Interface for Disorders of Execution [35, 36]) is a system that has been developed to provide natural, speech-based guidance and to allow user feedback. It aims to mimic the scaffolding provided by carers. GUIDE helps an individual through tasks using only verbal prompts and verbal feedback from the individual. The system issues audio prompts and obtains spoken responses, thus simulating natural dialogue. This type of interaction is familiar to the individual, so very limited learning is needed. There are no visual cues or prompts to draw the user's attention away from the task at hand. GUIDE also uses speech recognition to gain verbal feedback from the individual. This is in line with previous research [32] which suggested that useful assistive technology should not require manual feedback.

GUIDE is based on the idea that caregivers are expert 'assistants for cognition' [35]. This idea comes from developmental psychology, where an expert's verbal scaffolding of a task is conceptualised as directly augmenting the novice's cognitive function. GUIDE prompting dialogues are based on a close analysis of the actual prompts that caregivers provide when observed in real-world contexts.

Automating prompting dialogues has both limitations and benefits. The obvious limitation is that the system can provide only verbal prompts; the system cannot model or demonstrate an activity. This means that users should be able to perform the given task with only verbal prompting. Adding visual cues and demonstrations to the system, as used by COACH, would also be possible. The main benefit of automating prompting dialogues is that it removes the interpersonal dimension, such that users are less likely to feel dependent on someone else and less likely to experience the system as nagging.

Following this approach, prompting dialogues are produced with the following structure. A task is broken down into sub-steps. Each sub-step begins with an orienting prompt that simply states the sub-goal, aiming to focus the user's attention. The dialogue then proceeds through a series of checks which are posed as questions. Each question is meant to stimulate a self-regulatory process that helps the user to avoid common errors. The user can verbally respond to each check by saying yes or no. If the response is yes, the dialogue moves swiftly onto the next check or step. If the response is no, a problem-solving procedure is followed with questions and prompts.

The use of checks has two benefits. First, it positions the user as the expert as opposed to prompting systems that control the user. Second, it clearly partitions the dialogue into the main path and problem-solving 'side paths'. If the user encounters no problems, then the dialogue proceeds swiftly. However if problems are encountered, they are identified and additional prompts are provided.

GUIDE runs on a desktop or laptop computer. Users interact with the system through either a wireless headset or a wired microphone array. The wireless headset can be a compact earpiece (such as used with mobile phones) and thus not be intrusive for the user; alternatively, a full operator's headset can be used. These have the advantage of picking up minimal ambient noise, but they have to be kept charged and the user must remember to wear them. A wired microphone array gives better sound quality and requires no setup, but also picks up ambient noise. The computer receives speech input, processes it using Automatic Speech Recognition, and uses it to trigger appropriate prompts. The coordinating software and dialogues are written in Pure Data, a program-

ming environment for audio and media processing [69]. The best results are obtained when using an array microphone, audio filters that remove non-human sounds, and a reduced vocabulary. It is then possible to achieve 99% recognition accuracy in a natural context (one person in a room performing the task). An obvious limitation is that the system is not suited to noisy environments or where multiple people are speaking.

In one study, GUIDE was used to support eight amputees with cognitive impairment when putting on a prosthetic limb. The study found that there were significant reductions in both the number of safety-critical errors and the number of steps forgotten or missed [36]. Another study involving one participant with cognitive impairment showed that the individual adapted to the use of GUIDE in the first session. This is in line with the claim that GUIDE can be used with minimal learning, unlike some of the other devices discussed earlier. Further studies involving adults without cognitive impairment exhibited fewer mistakes and hesitations using GUIDE compared to written instructions, and that more positive comments were made about GUIDE [35].

The protocols (i.e. dialogues) developed during research on GUIDE try to emulate the verbal scaffolding support provided by carers. The dialogues have been thoroughly researched and evolved, based on consultations with occupational therapists, expert carers, physiotherapists and observations of users performing both assisted and unassisted tasks. The findings suggest that voice-mediated assistive technology for cognition can materially assist individuals and their carers to lead more independent lives.

The current GUIDE system does not have an easily used design tool for creating well-structured prompting dialogues. All speech output is pre-recorded rather than using TTS (Text To Speech). Because the protocols are tailored to individual patients and modified over time, the prompts are often recorded in diverse environments and can thus sound non-uniform. Moreover, there is a tendency for inconsistencies to arise within the protocols; these can become very complex and difficult to debug. Since there is no textual representation of the verbal aspects of dialogues, it is essentially impossible to perform any analysis on the prompts being provided. GUIDE also has no way of verifying dialogue completeness, correctness and consistency. As a result, users may have problems with hastily constructed protocols, with the dialogue entering an infinite loop or coming to an unexpected end.

In this paper, a new approach to dialogue design builds on the strengths of CRESS (Communication Representation Employing Systematic Specification, Section 2). The work has shown how CRESS can be used to design dialogues in a usable (graphical) manner, how it can automatically check dialogue integrity, and how it can automatically create dialogue implementations.

1.3.4. The LOTOS Formal Method

Formal methods are mathematically-based techniques for precise description and analysis of systems. A specification is an abstract and high-level description, whereas an implementation is a concrete and executable description. In software engineering, validation checks that a system meets its requirements (‘doing the right thing’), while verification checks that the system is being built properly (‘doing the thing right’) [3]. However in formal methods (and this paper), the use of these terms is different: validation means mathematically-based testing, while verification means mathematical proof that a system satisfies certain properties.

LOTOS (Language Of Temporal Ordering Specification [24]) is a internationally standardised language for formal specification and rigorous analysis. Although conceived for use with communications systems, LOTOS has been used in many other areas. As examples from the medical field, LOTOS has been used for modelling and testing radiotherapy accelerators [51], and for modelling and analysing clinical guidance trees [53]. LOTOS is classed as an algebraic specification language: abstract data types are specified by equations defining their operations, and behaviour is specified by interacting processes whose behaviour follows algebraic rules. Unlike a number of formalisms, LOTOS fully supports the integrated specification of data and behaviour.

LOTOS was chosen to model prompting dialogues for several reasons: its flexibility and expressibility, the prior work on translating interactive voice services into LOTOS, and the good support for analytic techniques and tools. An overview of LOTOS is given in [4], while online tutorials can be found at *www.inrialpes.fr/vasy/pub/cadp* and at *www.cs.stir.ac.uk/well*.

LOLA (LOTOS Laboratory [37]) is the tool that was used to validate prompting dialogues. LOLA has commands to generate the state space of a specification, subject to constraints such as limiting the exploration depth or combining the behaviour with a test process. CADP (Construction and Analysis of Distributed Processes, *www.inrialpes.fr/vasy/cadp*) is the toolset that was used to verify prompting dialogues. Efficient verification with CADP normally requires key data types to be coded by hand [18]. CADP also does not handle parameterised ('formal') types in LOTOS. However, the new work in this paper automates the entire procedure for analysis.

The authors are unaware of any work by others to mathematically model and analyse dialogues. There is, however, a standard approach to dialogue development using 'Wizard of Oz' experiments [25]. The idea is that the developer pretends to be the dialogue system while test users interact with it. This requires the developer to follow a dialogue script, though VoiceXML has been used to automate this process (*http://david.portabella.me/dialogue*). The procedure is useful for developing the design of a dialogue. However, it does not prove (in any mathematical sense) that a dialogue is free from undesirable errors such as dead-ends, unproductive loops, or failures to terminate as expected. It also does not prove that a dialogue exhibits desirable properties such as always booking an available flight or transferring money between bank accounts. This paper focuses on these kinds of issues in determining the correctness and consistency of dialogues.

1.3.5. The VoiceXML Scripting Language

VoiceXML [61] is a widely used scripting language for IVR (Interactive Voice Response). Although mainly used in automated telephony systems, VoiceXML also lends itself to prompting dialogues. VoiceXML treats a dialogue like filling in a form whose items are entered by responding to speech prompts. Because VoiceXML aims to be speaker-independent, the possible responses are tightly constrained by a grammar such as Boolean (yes/no responses). Once a form item has been completed, the next item is requested.

Each form item is associated with a variable that contains the user's response. There is also a prompt count that records how often a prompt has been issued. The reaction to an invalid response, say, can be made to depend on the prompt count (e.g. to give

up after a certain number of attempts). Besides forms and items, VoiceXML supports sub-dialogues (like subroutines), loops, branches, interaction with web applications and databases, and JavaScript.

VoiceXML is described in [43], while online tutorials can be found at *www.vxml.org*. There are several commercial implementations of VoiceXML such as Nuance Café (*www.nuance.com*) and Voxeo Prophecy (*www.voxeo.com*)

1.4. Overview of The Article

Section 2 discusses how dialogues in general, and prompting dialogues in particular, can be created with CRESS. The sample dialogues used in this paper are introduced. Section 3 explains how dialogue designs are analysed through automatic formal specification, validation and verification. Although formalisation is an optional step, it is important in establishing confidence in the dialogue design. Section 4 deals with the practical implementation and deployment of dialogues using a VoiceXML platform. Section 5 evaluates the approach from the perspectives of dialogue design and dialogue use, and also notes current limitations. Section 6 summarises the overall results and gives pointers to future work.

2. Modelling Dialogues with CRESS

This section gives an overview of the CRESS methodology for (voice) service design. Examples are given of dialogues that were developed to support people with cognitive impairment in completing daily tasks.

2.1. CRESS Methodology

CRESS (Communication Representation Employing Systematic Specification) is a graphical notation for describing the flows in services, a methodology for service development, and a comprehensive toolset (*www.cs.stir.ac.uk/~kjt/research/cress.html*). Currently CRESS handles services in seven different domains, and supports code generation for five different languages. The foundational work in [47] introduced a notation for telephony features. This was subsequently considerably adapted and extended to describe Internet telephony services [48], IVR services [49], web services [50], grid services [57], device services [54] – and now prompting dialogues. The service development methodology has recently been rounded out with capabilities for convenient formal verification and implementation evaluation. Relative to previous publications on CRESS, this paper covers the complete methodology with a new application to prompting dialogues.

Dialogues are described manually using the CRESS graphical notation. Several graphical editors can be used, but the preferred one is CHIVE (CRESS Home-Grown Interactive Visual Editor, *www.cs.stir.ac.uk/~kjt/software/graph/chive.html*). Diagrams can be automatically translated into formal (i.e. mathematically precise) specifications. The core CRESS notation is independent of the application domain and target languages. In this paper, formal analysis of prompting dialogues is achieved through automatic translation to the LOTOS formal specification language.

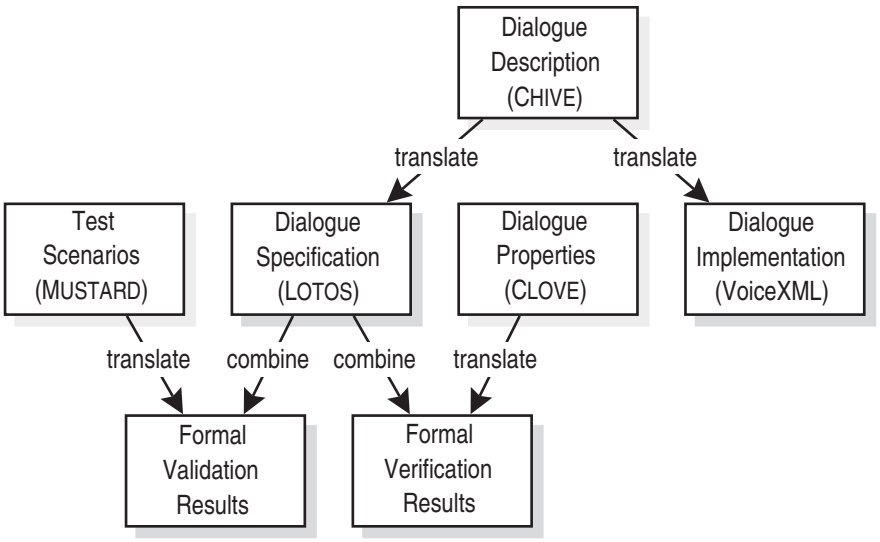

Figure 1: CRESS Methodology for Dialogue Design

The CRESS methodology is shown graphically in Figure 1. Later sections illustrate the methodology for creating dialogues to support people with cognitive impairment. The dialogue designer begins by creating a graphical dialogue description using CHIVE. CRESS offers a thorough approach to checking dialogues – particularly for when they are large or complex. It is therefore recommended to first analyse the dialogue using a variety of formal checks: validation and verification.

Formal validation of dialogue specifications is convenient and quick. It copes with large (even infinite) state spaces. As a form of testing, validation is necessarily incomplete. However, it complements what is possible through verification. The LOTOS specifications generated by CRESS can be used immediately for formal validation.

Test scenarios are created manually using the MUSTARD language (Multiple-Use Scenario Test and Refusal Description, *www.cs.stir.ac.uk/~kjt/research/mustard.html*). MUSTARD is a high-level language for expressing tests independently of the application domain and the target language [52]. In this paper, dialogue tests are automatically translated into LOTOS and formally validated. Validation results are presented in MUSTARD terms so that the user does not need to be familiar with the underlying formalism or tools.

Formal verification is more challenging, but allows general properties of a dialogue to be checked – not just particular scenarios as with validation. Properties that a specification should respect are defined manually using CLOVE (CRESS Language-Oriented Verification Environment, *www.cs.stir.ac.uk/~kjt/research/clove.html*). CLOVE supports the high-level description of desirable properties that a system should exhibit [58]. CLOVE is independent of the application domain and the target language. Certain properties are automatically checked by CLOVE, e.g. freedom from deadlock (where progress halts), freedom from livelock (unproductive internal loops), and guaranteed

termination (successful completion). Verification results are presented in CLOVE terms so that the user does not need to be familiar with the underlying formalism or tools.

In this paper, dialogue properties are automatically translated into μ-calculus [6] and model checked (a general technique that shows a specification respects certain properties [31]). μ-calculus is a logic that allows behavioural properties to be defined. CLOVE makes use of techniques such as on-the-fly verification (generating states as required) and compositional verification (piece-by-piece). However, state space explosion often limits what is practical (a problem that is common to all state-based verification techniques).

The result of validation and verification is a dialogue description in which the developer can have a high degree of confidence. The final step is automatic generation and deployment of operational code. For prompting dialogues, this involves creating VoiceXML. If the developer is confident in the design of a dialogue, it is possible to omit formal validation and verification. The dialogue can then be immediately translated into VoiceXML and deployed for use. However there can be more confidence in the dialogue design if it has been formally analysed beforehand.

2.2. CRESS Notation

A CRESS diagram is a directed graph that shows the flow of actions in a dialogue; examples appear later in Figures 2 and 3. CRESS dialogues deliberately follow the principles of VoiceXML. The subset of CRESS activities appearing in this paper is explained in Table 1. For dialogues in general, CRESS supports a much richer range of constructs than is described here. For example, dialogues can deal with a wide variety of user responses, event guards, dialogue-defined events at multiple levels, configurable reprompting, and flexible data handling [49].

For people with cognitive impairment, it would be very undesirable to have complex prompts and options. As argued in Section 1.3.1, a scaffolding approach with simple requests and answers is much more appropriate. As a result, CRESS dialogues for people with cognitive impairment make very restricted use of dialogue constructs.

In a CRESS diagram, numbered nodes (ellipses) define actions that exchange information with the user or are internal to the dialogue. Along the arcs that define dialogue flow, expression guards (e.g. yes, no) or event guards (e.g. NoInput, NoMatch) determine whether a path is followed. Although not used in this paper, a CRESS rule box (a rounded rectangle) defines things like variables, macros and use of subsidiary diagrams. Multi-page diagrams can be created, using connectors (plain text labels) to link different parts of a diagrams.

2.3. Dialogues for People with Cognitive Impairment

Four sample dialogues were studied for the work in this paper. These dialogues support users who also have some form of cognitive impairment. Since many target users will be older people, comorbidity is likely (e.g. diabetes coupled with dementia).

Glucose: This guides someone with diabetes through the process of checking blood sugar level. The CRESS dialogue was closely modelled after the one developed for the GUIDE prompting system [36].

Construct	Meaning
Audio *"message"*	This outputs a speech message to the user.
Catch *"event ..."*	This defines how to handle the specified events. Standard events include Cancel (user-requested cancellation), Exit (user-requested termination), Help (user-requested help), NoInput (no user response) and NoMatch (invalid user response).
Clear *"variable ..."*	This clears the specified variables, allowing the corresponding requests to be issued again. Query variables are simply identified by their node numbers.
Exit	This terminates the dialogue normally.
Query *"prompt"*	This prompts the user to respond with yes or no. It is a shorthand for a request with a Boolean result, followed by a check for a yes/no response.
Reprompt	This causes the most recent prompt to be repeated.
Start	Used to indicate the start of a diagram if this would otherwise be ambiguous.

Table 1: Subset of CRESS Dialogue Constructs

Handwash: This helps a person through the process of hand washing. The CRESS dialogue was adapted from the one developed for COACH [34].

Limb: This guides an amputee through the process of donning a prosthetic limb. Again, this is a GUIDE example adapted for CRESS.

Smoothie: This guides a person through the process of making a strawberry smoothie. Again, this is a GUIDE example adapted for CRESS.

The dialogues were chosen as illustrative of the kinds of tasks that people with cognitive impairment may need help with: medical procedures (Glucose, Limb), bathing (Handwash) and food preparation (Smoothie). The approach is also appropriate for other daily activities such as dressing, housework, making appointments, using domestic appliances, and route planning. In all four cases, the source material in textual form was converted into CRESS dialogue diagrams. These were then formally validated (section 3.2), formally verified (section 3.3) and evaluated with end users (section 5.2).

Table 2 presents various statistics about the sample dialogues in order to give some idea of their scale. The table gives the number of nodes (i.e. actions) in each CRESS dialogue diagram, and the number of lines of code in the LOTOS specification and the VoiceXML implementation. In terms of size, these are non-trivial dialogues. From a LOTOS point of view, the specifications are fairly large. For comparison, LOTOS specifications have been written and analysed of a file system (1150 lines [39]), an invoicing system (180 lines [56]), the design of a CPU (1450 lines [55]), and a digital phone network (1760 lines [14]).

Dialogue	Diagram (nodes)	LOTOS (lines)	VoiceXML (lines)
Glucose	98	12,417	918
Handwash	32	1,421	347
Limb	112	5,528	1,385
Smoothie	196	39,066	2,170

Table 2: Sample Dialogues for People with Cognitive Impairment

The diagrams, specifications and implementations are all too large to present in this paper; only selected extracts are therefore given. However, the complete set of files has been made available for download (see section 3.1). To give a concrete idea of what the dialogues look like, extracts from the Limb example are given in Figures 2 and 3.

Figure 2 shows the first step of the limb-donning dialogue. Global event handlers are defined at the top level for situations such as the user saying nothing or exiting the dialogue. The Help connector (different from the Help event) is reached from other parts of the diagram. Suppose the user answers no to the query in node 103, no in node 104, yes in node 105, and no in node 108. After node 110, the dialogue repeats so that the user has the opportunity to go through the questions again. The default dialogue rules mean that the user would not be reprompted for these queries because they have already been answered. The Clear in node 110 therefore removes previous responses so the user can answer the queries again. In fact, this allows subtle control over how a dialogue behaves where there are loops.

Figure 3 shows a later stage of the dialogue where the user is asked to remove the footplates from the special chair they are sitting in. This time there are two loops back to earlier questions, so two Clear actions are required (nodes 226 and 228).

3. Analysing Dialogues with CRESS

This section explains the automatic formal specification, formal validation and formal verification of dialogues. The results of formal analysis are discussed.

3.1. Automatic Specification

The Check menu option in the CHIVE diagram editor ensures correct dialogue syntax. The Validate and Verify menu options are used to check diagram semantics via automatic translation into a LOTOS specification.

Since LOTOS is a specialised language, sample code is not given here. In any case, the point of CRESS is that the dialogue designer never needs to see the underlying specification. The interested reader can, however, find the dialogue specifications in *www.cs.stir.ac.uk/~kjt/software/download/ivr-examples.zip*. The specifications do not, of course, use actual speech – only the textual equivalent of this. Each dialogue query corresponds to a LOTOS process (somewhat like a subroutine). Process parameters include the current dialogue prompt count and a history of previous query answers.

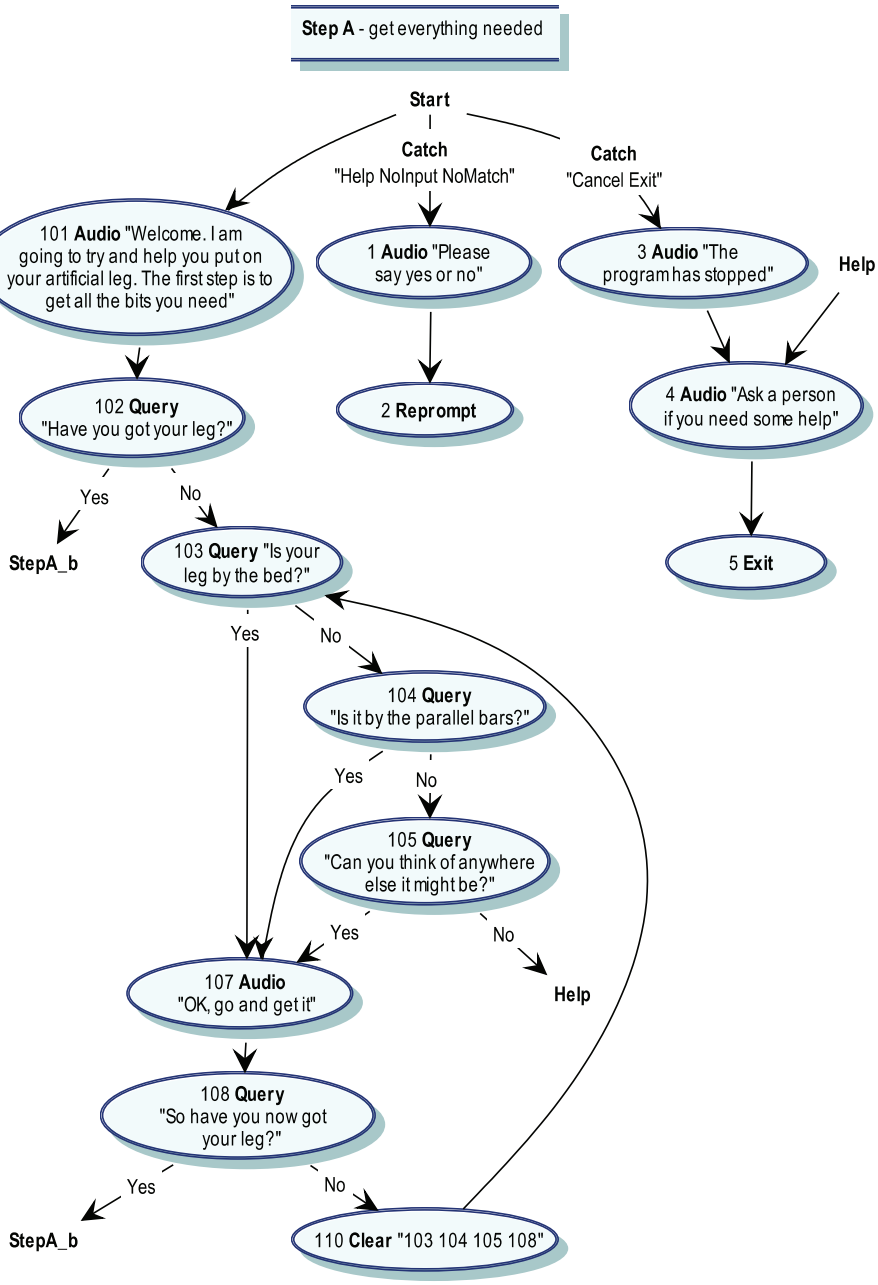

Figure 2: Limb Donning Dialogue Step A

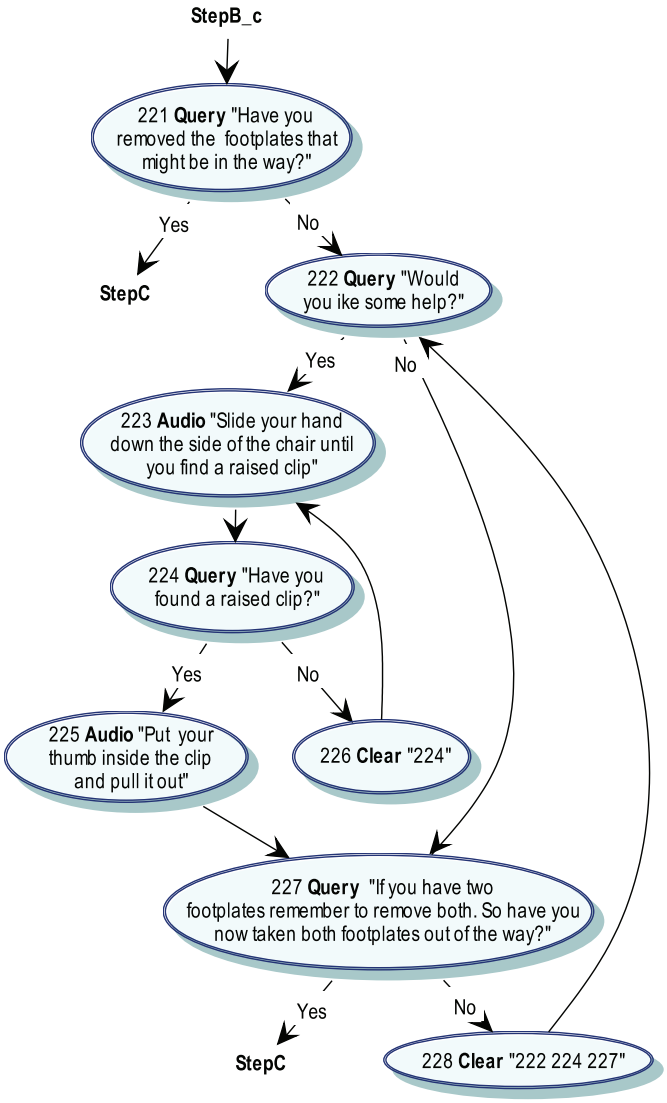

Figure 3: Limb Donning Dialogue Step B, Part c

Construct	Meaning
agree(*prompt*)	This expects the dialogue to speak the given prompt and then to hear the user reply yes.
deny(*prompt*)	This expects the dialogue to speak the given prompt and then to hear the user reply no.
hear(*message*)	This expects to hear the given message from the dialogue.
succeed(*behaviour*, ...)	This requires the sequence of behaviours to complete successfully.
test(*name*, *behaviour*)	This defines the name and behaviour for a test.

Table 3: Subset of MUSTARD Test Constructs

Besides processes, the LOTOS translation includes dialogue-specific data types and event dispatching code (automatically created according to the diagram content). The specification is supplemented by a substantial (but shared) data type library for CRESS.

3.2. Automatic Validation

Formal validation is performed on the automatically generated specification. Test scenarios are written using the MUSTARD language (Multiple-Use Scenario Test and Refusal Description,*www.cs.stir.ac.uk/~kjt/research/mustard.html*) which was introduced in Section 2.1. The subset of MUSTARD constructs used in this paper is summarised in Table 3.

MUSTARD supports a very much richer range of test constructs than used here for dialogues [52]. For example, MUSTARD also supports test fixtures (predefined parts of tests), acceptance and refusal tests (what must and must not happen), deterministic and non-deterministic tests (decisions made by the test or the system), sequential and concurrent tests (linear or parallel), tests that depend on the presence of features, and tests that manipulate variables. However, the nature of prompting dialogues for people with cognitive impairment means these more sophisticated capabilities are not required.

MUSTARD also supports domain-specific test definitions such as for dialogues. In fact, the **agree**, **deny** and **hear** actions are defined simply using more basic primitives. This flexibility makes MUSTARD easy to tailor for new applications.

Formal validation of the Limb dialogue is used as an example. The designer selects Validate to see the results in Figure 4. Each test states the dialogue name, the test name, and whether it passes validation. The CPU time to perform each test is also shown (for a 2.67 GHz processor).

It is well known from software testing that programmers should not be asked to test their own code. This is partly because human nature could encourage the tester to confirm that the code is correct, not to find errors. More importantly, tests written by the programmer could well repeat the same misconceptions that have been coded into the program. A similar practice was adopted when validating the dialogues described in this paper. Because the authors are in transition from GUIDE to CRESS, it was necessary to start with GUIDE dialogues that had been developed manually. Future

```
Generating tests ...
Generating tests for Limb
Running tests ...
Test Limb No Problems ...                Pass    4.8 secs
Test Limb Help Needed ...                Pass    1.8 secs
Test Limb Find Limb ...                  Pass    1.8 secs
Test Limb Apply Brakes ...               Pass    2.0 secs
Test Limb Remove Boards ...              Pass    2.0 secs
Test Limb Remove Footplates ...          Pass    2.4 secs
...
```

Figure 4: Extract from Formal Validation of Limb-Donning Dialogue

dialogues will be developed directly from CRESS, thus eliminating one step in what was done for this paper.

To determine the suitability of the CRESS approach, one author (McMichael) turned the GUIDE textual dialogues into CRESS form. Independently, a second author (Turner) wrote test scenarios based on an understanding of what the dialogues were meant to do. These test scenarios were then applied to the dialogue specifications. The kinds of issues found are discussed in Section 3.4.

Most test scenarios have a similar structure that exercises critical paths through a dialogue to make sure it behaves as expected. As a concrete example of what a test looks like, consider part of the Limb dialogue in Figure 3. This deals with removing chair footplates prior to putting on the artificial limb. The test in Figure 5 exercises this part of the dialogue. Initially, the queries are answered positively. The user then says that the footplates have not been removed (node 221) and agrees that help is needed (node 222). When the raised clip is not found (node 226), the user is asked again find this (nodes 223 and 224). Once the clip has been found, the user is asked to pull it out (node 225). If the user has still not removed the footplates, the relevant part of the dialogue is repeated (from node 222). After this, the user confirms footplate removal.

A total of 62 test scenarios (27 for Limb) were written for the four dialogues introduced in Section 2.3. The number of exchanges with the user in each test varied from 12 to 202. In fact, only a small number of tests were written of the Smoothie dialogue. This was partly because it is not safety-critical. More interestingly, a different test strategy was adopted for this dialogue: all 'side paths' dealing with problem solving were exercised in a single, very lengthy test. For the other dialogues, smaller and more modular tests were created.

As an alternative, it would have been possible to check the dialogues by translating them to VoiceXML and then trying to exercise all important paths. However, this is a very tedious and error-prone approach that is hard to repeat reliably. (Notwithstanding this, it is how IVR dialogues are usually tested.) This strategy also does not provide concrete evidence that a dialogue has been adequately tested.

In contrast, formal validation provides a repeatable way of checking a dialogue, and also serves as evidence of exactly what has been tested. Apart from initial validation, the test scenarios have a useful other purpose. Most dialogues will go through various stages of evolution as they are tried with users. If the dialogue evolves, a conventional

test(Remove_ Footplates,
  succeed(
    hear(Welcome. I am going to try and help you put on your artificial leg.
        The first step is to get all the bits you need),
    agree(Have you got your leg?),
    agree(Have you got your liner?),
    agree(Have you got your socks?),
    hear(In this step you will be securing your chair),
    agree(Have you got both brakes on?),
    agree(Have you removed any stump boards that might be in the way?),
    deny(Have you removed the footplates that might be in the way?),
    agree(Would you like some help?),
    hear(Slide your hand down the side of the chair until you find a raised clip),
    deny(Have you found a raised clip?),
    hear(Slide your hand down the side of the chair until you find a raised clip),
    agree(Have you found a raised clip?),
    hear(Put your thumb inside the clip and pull it out),
    deny(If you have two footplates, remember to remove both.
        So, have you now taken both footplates out of the way?),
    agree(Would you like some help?),
    ...
    agree(If you have two footplates, remember to remove both.
        So, have you now taken both footplates out of the way?),

Figure 5: Test Scenario for Footplate Removal

(Wizard of Oz) check would be time-consuming to repeat (and might be difficult to repeat exactly). In contrast, formal validation acts as an ideal regression test. Only the parts of a dialogue that have changed need to be modified in the test scenarios. In a rehabilitation context, full manual testing of dialogues would be unlikely to be feasible for therapists. However automated testing (formal validation) would be practicable for, say, a therapist adjusting a dialogue in someone's home.

3.3. Automatic Verification

Even though the validation just discussed is formally based, it has two advantages. Firstly, it is practical even if the specification has an infinite state space because a test limits behaviour to a concrete scenario. Secondly, it follows the kinds of principles used in software testing and so is familiar. However, validation is only for specific test cases and does not prove things in general about a specification. For this reason, formal verification is a useful complement to validation as it aims to prove generic properties. The snag is that state-based verification requires a finite (and practicably small) state space. Given this, model checking (i.e. proving properties) is a viable 'push button' form of verification.

In all state-based verification, it is common to find that restrictions on the specification are necessary. As a typical example, [56] describes the verification of an invoicing system that uses reference numbers, product codes and order quantities (all non-negative integers). Model checking is viable only if these three kinds of num-

bers are limited to the values 0 and 1 – a tiny range. Even a value of 2 gives rise to excessively large state spaces.

The four dialogues treated in this paper were verified only after their specifications had been restricted. Interestingly (and unusually for verification), these restrictions leave the specification behaviour *unaltered*. The same restrictions can also be applied to any dialogues of this nature, so the approach is generally applicable.

The first restriction limits user responses to yes and no. This is done by restricting the range of messages (text) considered – a part of the CLOVE definitions for verification. At first it would appear that this would not check error handling, e.g. for absent or invalid responses. However, dialogue specifications in LOTOS allow user events such as NoMatch and NoInput that are exercised during verification. Even with this restriction, the smallest dialogue considered here (Handwash) exceeds the CADP verification tool limit: 2^{32} (4.3 billion) states or transitions on a 32-bit processor.

The second restriction takes into account that CRESS dialogues support something that is not used in dialogues for people with cognitive impairment. Every form item has an associated variable; these are implicit for Query nodes, but part of the specification. This allows the later part of a dialogue to make use of the answer to an earlier (query) field. As someone with cognitive impairment may well have limited short-term memory, the dialogues do not use this feature.

The consequence is that the history of responses is not in practice useful for these dialogues (though it can be used in more general dialogues). There is only the requirement to store a response temporarily so that it can be checked against yes or no. It is therefore sufficient to hold a single query response. The CLOVE tool has an option for restricting the size of data structures such as the query history, so this is easily set to 1. This has a dramatic effect on the size of the state space. Handwash, for example, changes from being practically unverifiable to having 97,653 states and 56,5210 transitions. However, even this restriction is not sufficient. The restricted Limb dialogue still has 561,770 states and 3,252,002 transitions – just at the limit of being verifiable with CADP. The most complex dialogue (Smoothie) still breaks the state space limitations.

The third restriction relates to the prompt count that every form field has. In dialogues with loops (like Limb and others), this prompt count is incremented without bound if prompts are repeatedly re-issued. The result is that the state space is infinite. CRESS (and VoiceXML dialogues in general) often make use of the prompt count. For example, a more detailed prompt may be given after a couple of invalid responses, or the dialogue may terminate if there are too many incorrect answers. This capability would be useful in dialogues for people with cognitive impairment, though the dialogues studied so far do not do this. Instead the dialogues take advantage of the fact that a person will give up after being asked the same question several times. All the dialogues considered here have an ‘escape route’ that allows the user to stop and ask for someone’s help.

The consequence is that the prompt count is not needed by these dialogues. Fortunately, the LOTOS translator has an option to suppress use of a prompt count. This dramatically reduces the size of the state space again. For example, the Limb dialogue then has only 753 states and 4,210 transitions. Even the Smoothie dialogue reduces to 6,853 states and 39,066 transitions. These are completely manageable and allow verification to take place.

Construct	Meaning
inevitable(*signal*, ...)	The given pattern of signals must occur along *all* paths in the dialogue.
initials(*signal*, ...)	This defines the initial signals that a specification should accept.
literals(**strings**, *text*, ...)	This lists the text values that should be considered during verification.
or(*signal*, ...)	This defines alternative patterns of signals.
possible(*signal*, ...)	The given pattern of signals must occur along *some* path in the dialogue.
property(*name*, *definition*)	This defines the name and property that the specification must respect. Spelled **Property**, this construct defines a property that must *not* hold.
sequence(*signal*, ...)	This defines a sequence of signals in a dialogue. Spelled **Sequence**, this construct allows internal specification actions between observable signals.

Table 4: Subset of CLOVE Property Constructs

Formal verification is performed on the automatically generated specification. Dialogue properties are written using the CLOVE language (CRESS Language-Oriented Verification Environment, *www.cs.stir.ac.uk/~kjt/research/clove.html*) which was introduced in Section 2.1. The subset of CLOVE constructs used in this paper is summarised in Table 4. In this context, a signal is an utterance by the user or the dialogue.

CLOVE supports a richer range of property definitions than is required for verifying the dialogues in this paper. For example, it also supports enumeration of various kinds of data types and structures, patterns of behaviour, and their logical combinations [58]. CLOVE, like MUSTARD, supports domain-specific definitions: **agree**, **deny** and **hear** have not been repeated here from Table 3. As for validation, verification properties were defined by one author (Turner) independently of the dialogue descriptions (McMichael). The kinds of issues found are discussed in Section 3.4.

Formal verification of the Limb dialogue is used as an example. The designer selects Verify to see the results in Figure 6. Each check states the dialogue name, the property name, and whether it passes verification. The CPU time and elapsed time to check each property are also shown (for a 2.67 GHz processor). The elapsed time is noticeably longer for generating the specification state space. This is because the procedure is input-output limited rather than processor limited.

Specifications are often verified to be free from deadlocks (where progress halts) and livelocks (unproductive internal loops). It is also useful to check that a specification starts out as expected by handling the utterances allowed by **initials**. In fact the dialogues in this paper are designed to terminate, so a check for deadlock freedom is pointless as the dialogue will definitely stop. Instead, a subtler check is required: that a dialogue exits normally. This verifies that a dialogue does not reach a dead

Generating properties for Limb ...		CPU Time	Real Time
Generating state space for Limb ...	Success	24.5 secs	10.0 mins
Verifying Limb Always Exit ...	Success	6.4 secs	7.0 secs
Verifying Limb Initials Safety ...	Success	6.6 secs	7.0 secs
Verifying Limb Livelock Freedom ...	Success	6.4 secs	10.0 secs
Verifying Limb Can Finish ...	Success	6.5 secs	7.0 secs
Verifying Limb May Not Finish ...	Success	6.6 secs	7.0 secs
Verifying Limb Finish Or Help ...	Success	6.4 secs	7.0 secs
Verifying Limb Remove Footplates ...	Success	6.3 secs	13.0 secs
...			

Figure 6: Extract from Formal Verification of Limb-Donning Dialogue

end, and also that it does not become stuck in a loop. The 'Always Exit', 'Livelock Freedom' and 'Initials Safety' properties are generic and built into CLOVE. They are therefore invoked through a tool option rather than requiring definition as properties. In fact, verifying just these properties may give sufficient confidence without formulating more dialogue-specific ones.

Figure 7 gives concrete examples of what verification properties look like for limb donning. The specification must start with the user hearing the welcome message (**initials**). Text values for verification are yes and no (**literals**). The *Can_Finish* property says that it is possible to reach the final congratulation message. The *Finish_Or_Help* property says that two outcomes are inevitable: either the final congratulation message is heard, or the user is told to ask a person for further help. The *Remove_Footplates* property resembles the *Remove_Footplates* scenario in Figure 5. The key difference is that the property is checked anywhere in the dialogue, whereas the scenario requires a particular preamble that leads up to the part of the dialogue of interest. As a result, the property focuses on the important part of the dialogue and therefore does not need extraneous description.

3.4. Results of Formal Analysis

All the dialogues studied in this paper had already been thoroughly checked on the GUIDE and COACH projects. For example, the limb-donning dialogue from GUIDE had already been through 17 stages of refinement and had been evaluated through clinical trials. Errors (particularly deadlocks) were frequently found during the original dialogue development; some of these emerged only during trials. As a result, the GUIDE developers recognised the need for more automated checking. In view of the extensive prior work on dialogue design, it was not expected that formal analysis would find much wrong with the dialogues. The situation will be different in future, however, when new dialogues are created from scratch. Then the rigour of formal validation and formal verification will be very useful.

While developing the dialogues in CRESS, the tools discovered syntax errors that were the result of transcription problems (e.g. unconnected nodes or queries not followed by 'yes' and 'no'). However, more interesting problems were found by the semantic checks:

Dialogue Style Although CRESS does not yet perform stylistic checks on the dialogue content, the formal analysis nonetheless found inconsistencies. (Stylistic check-

```
initials(
  hear(Welcome. I am going to try and help you put on your artificial leg.
       The first step is to get all the bits you need))

literals(strings,
  yes,no)

property(Can_ Finish,
  possible(
    hear(Well done! You have now put your leg on safely)))

property(Finish_ Or_ Help,
  inevitable(
    or(
      hear(Well done! You have now put your leg on safely),
      hear(Ask a person if you need some help))))

property(Remove_ Footplates,
  possible(
    Sequence(
      deny(Have you removed the footplates that might be in the way?),
      agree(Would you like some help?),
      hear(Slide your hand down the side of the chair until you find a raised clip),
      agree(Have you found a raised clip?),
      hear(Put your thumb inside the clip and pull it out),
      agree(If you have two footplates, remember to remove both.
            So, have you now taken both footplates out of the way?))))
```

Figure 7: Sample Verification Properties for Limb Donning

ing will be automated in a future version of the approach.) The test scenarios and verification properties reflected what the dialogues were expected to say, not what the CRESS diagrams actually said. As a result, a number of failures were encountered and corrected, arising from small editorial inconsistencies.

As an example, 'Can you put the liner on?' vs. 'Can you put on the liner?' are equivalent but unnecessary variations in a dialogue that might confuse people with cognitive impairment. The original dialogues were also found to use pronouns frequently, e.g. 'Is it too tight?' rather than 'Is the liner too tight?'. Since the dialogues are taken slowly in practice, someone with cognitive impairment could easily miss the referent. The scenarios and properties were formulated as consistent and unambiguous statements of what the dialogues should do. It was only when these were checked against the CRESS descriptions that differences were found.

Dialogue Links The dialogues have frequent links to connectors elsewhere in their diagrams. For the largest dialogue (Smoothie), it was found that in some cases the CRESS diagram branched to the wrong place (reflecting an error in the original GUIDE description). Fortunately this dialogue was not a safety-critical one (and

had not been previously checked by the GUIDE developers as thoroughly as the others). Nonetheless, such errors are easy to make in dialogue design and could have undesirable consequences.

Clearing Answers Some instances were found of incorrectly using Clear to remove previous answers. The effect was that part of a dialogue would not repeat correctly. Although Clear provides subtle control over repeating dialogues, its use for this work is actually unnecessary as CRESS could automatically determine what is being repeated. This will be done in a future version of the approach.

Although the errors found were fairly minor, the new methodology has demonstrated that it can check correctness and consistency of previously well-debugged dialogues. There is therefore confidence that it will be useful on new dialogues.

4. Deploying Dialogues with CRESS

The final stage of dialogue development (implementation) is straightforward and automated. Indeed, this allows the developer to put effort into the important area of dialogue design rather than coding. Once the dialogue design has been thoroughly checked, the Realise menu option in the CHIVE diagram editor translates a diagram into VoiceXML and automatically deploys it. Since VoiceXML is a specialised language, sample code is not given here. In any case, the point of CRESS is that the dialogue designer never needs to see the underlying implementation. The interested reader can, however, find the dialogue implementations in *www.cs.stir.ac.uk/~kjt/software/download/ivr-examples.zip*.

The implementation work in this paper used V-Builder (VoiceXML engine, Automatic Speech Recognition) and Vocalizer (Text To Speech) from the Nuance Corporation (*www.nuance.com*). This allows the user to interact with the dialogues using a wireless headset and microphone. Future work may use separate Automatic Speech Recognition and Text To Speech packages, e.g. those developed by CereProc (*www.cereproc.com*) with whom the authors have collaborative links.

5. Evaluation

This section gives a preliminary evaluation of the methodology from the perspectives of dialogue design and dialogue use. Limitations of the approach are also noted.

5.1. Evaluating Dialogue Design

The dialogues considered in this paper were based on the work of others on the GUIDE and COACH projects. These dialogues had already been thoroughly developed and had been used in trials with end users. It was therefore not necessary for the authors to evaluate the efficacy of the dialogue designs. Rather, the new approach needed to be evaluated. It is expected that other prompting systems (e.g. PocketCoach) would also benefit from the work of this paper.

Designing voice services graphically is a key part of the CRESS methodology. At this stage in the evolution of the methodology, it was felt that people with software design experience would be a meaningful evaluation group. A mixed empirical evaluation

was performed to test the following hypothesis: someone with experience of software development, with 45 minutes of training on the approach and the CRESS system, can define small services, with 80% accuracy, in at most 15 minutes per service.

The authors recruited five software developers who had no previous experience of CRESS. The participants were given written instructions to follow in their own time, without training or advice from the authors. A copy of the CHIVE diagram editor was provided for local installation, along with a 'palette' of typical symbols used in constructing services. The instructions began with a three-page explanation of the approach and the CHIVE editor, including three diagrams that the participants were asked to study and then to reproduce themselves using the diagram editor. 45 minutes was suggested as appropriate for this phase, though no time limit was imposed.

The participants spent an average of 34 minutes (range 10 to 60) on the familiarisation phase. This compares favourably with the authors' expectation of 45 minutes. The shortest period (10 minutes) may reflect this participant's preference for learning by doing rather than extended prior study.

In the next part of the instructions, the participants were given five specific tasks to perform. Each task required a service diagram to be drawn (somewhat different from the examples), based on a natural language description. The participants were asked to record how long tasks took, and to save their diagrams on completion (or after 15 minutes if a task was not completed). The participants were asked to rate five statements about the approach on a five-point Likert scale. They were also given the opportunity to provide a free-form qualitative evaluation of the exercise.

Overall, participants completed tasks in an average of 5.7 minutes each, with an average accuracy of 88% (compared to the hypothesis of 15 minutes and 80%). The participants were asked to rate five statements about the approach on a scale from 1 (strongly disagree) to 5 (strongly agree):

Statement 1: *I was able to create the service diagrams without too much difficulty*: average score 3.8 (range 3 to 4).

Statement 2: *I found it fairly straightforward to translate the English descriptions into diagrams*: average score 3.2 (range 1 to 4).

Statement 3: *I found it fairly straightforward to create and edit diagrams using the diagram editor*: average score 3.6 (range 3 to 4).

Statement 4: *I think the approach would be usable by people with experience of software development*: average score 4.0 (range 3 to 5).

Statement 5: *I think that the approach could be useful in practice for defining services*: average score 3.2 (range 2 to 5).

The rating of statement 1 suggests that the approach is usable, though the diagram editor would benefit from some technical improvements. The authors had expected statement 2 to be least agreed with, since significant mental effort is required to translate a natural language description into any precise representation. Like statement 1, the scoring of statement 3 offers encouragement – though improvements to the diagram editor are desirable. The evaluation of statement 4 suggests that software developers

at least can use the approach effectively. Based on the accompanying free-form comments, the lack of a more positive response to statement 5 appears to reflect the need for improvements in the diagram editor rather than doubt over the general approach.

Given the short time that participants spent in familiarisation (average 34 minutes), their performance impressed the authors. Although the limited number of participants does not allow statistically valid conclusions, the results of the preliminary evaluation are encouraging. After improvements in the usability of the diagram editor, the evaluation will be repeated with the intended designers: care professionals (e.g. therapists).

5.2. Evaluating Dialogue Usability

The end results of development (the dialogue implementations) were also evaluated with five non-technical users without cognitive impairments. The aim was to assess usability rather than utility of the technology. Users were asked to follow through and interact with each of the four dialogues in a lab setting. This was accompanied by a mixed empirical evaluation of how well the users understood the dialogues and how comfortable they were with the technology.

As noted in section 3.4, the CRESS dialogues were closely based on dialogues from the GUIDE and COACH projects that had already been thoroughly checked. The usability evaluation thus reflected more on the VoiceXML platform than on the design methodology. Among the qualitative information collected, user opinions included:

- Two users felt that the dialogues were too fast and did not allow enough time between prompts. Although some control of speech delivery is possible with VoiceXML, in fact it lacks sufficient flexibility in this area. (GUIDE handles this through addition of pauses and control of playback speed.) The speech tools are completely independent of CRESS, so alternatives will be considered.
- Two users felt embarrassed about using the prompting system in a lab setting (where others were present). In fact the planned location for use is the home, where this is less likely to be an issue.
- All five users strongly agreed that they understood what the dialogues were asking them to do. They felt that the system was easy and natural to use, and required minimal learning.

Colleagues of the authors have demonstrated that they can learn to formulate test scenarios and dialogue properties. However, this aspect of the methodology is sufficiently new and different that it has not yet been evaluated with care professionals. This is planned as part of the ongoing work on the GUIDE project. It is anticipated that these users will be capable of formulating test scenarios, as the ability required is very similar to that needed to create dialogues in the first place. However, it is acknowledged that formulating dialogue properties may be more difficult for this group – though verification of generic properties (which do not require definition) coupled with formal validation is likely to be sufficient.

As far as the authors are aware, the CRESS approach to rigorous dialogue development is unique. Where standard techniques such as ‘Wizard of Oz’ exist, these are complementary to CRESS and do not emphasise the same design aspects.

5.3. Limitations of The Approach

Although the authors believe that the approach is general-purpose, it does of course have limitations:

Nature of Dialogues: The target of the work has been interactive, speech-based dialogues to support people with cognitive impairments in performing everyday tasks. The incremental, step-by-step dialogues developed so far could seem tedious to an unimpaired user. Complex tasks (e.g. choosing an investment or planning a holiday) would also be inappropriate for the target group. However, this is a reflection on the dialogue design principles that have been adopted (section 1.3.1). There is nothing in the technical methodology or the technology to limit its application. However, it would be fair to say that new applications would need to lend themselves to a fairly linear style of dialogue because of its speech-based nature.

Dialogue Design: The CHIVE editor currently used for designing dialogues is intended for many kinds of services. As a result, it is not sufficiently convenient or specialised for prompting dialogues.

Linguistic Analysis: Currently, CRESS does not perform any kind of linguistic analysis on dialogues. This would be desirable to ensure consistency and clarity of the dialogues, as well as conformance to good design principles.

Formal Aspects: Formal validation can be carried out on very complex dialogue specifications. However, this requires the designer to be willing to formulate test scenarios using MUSTARD. With limited training, this is feasible but needs effort.

Formal verification is likely to remain a specialised task since formulating desirable properties of a system requires particular thinking. However, a range of properties is already checked automatically without designer intervention.

Speech Technology: The methodology and the technology are able to support much richer dialogues, e.g. allowing a wider range of speech responses and more complex dialogue flows. For people with cognitive impairment, the current restrictions on dialogues are believed to be appropriate. Allowing freer use of spoken responses would lead to more frequent errors in speech recognition, and thus risk confusing someone who is already likely to be struggling. Similarly, more complex dialogues (e.g. the so-called mixed-mode dialogues of VoiceXML) would almost certainly risk the user getting lost.

Speaker-independent speech recognition is preferable as the system then does not need training for each user. However, this is challenging and requires restrictions on vocabulary and context. All speech technologies find it difficult to deal with environments that are noisy or where several people are speaking. These are challenges that the speech recognition community are working on. Since speech technology is an adjunct to the methodology of this paper, it is sufficient for the authors to take advantages of new developments as they become available.

Evaluation: GUIDE and COACH have been carefully evaluated with therapists and live users. However, CRESS has so far received only a preliminary evaluation.

Real-World Deployment: As present, the CRESS toolset is a research prototype. Although the tools are mature and robust (having been under development for a dozen years), they are not currently packaged up in a convenient way. As a result, installation requires specialised expertise.

In contrast, the authors believe that applicability and scalability are *not* in fact limitations. The approach has been shown to work on significant prompting dialogues developed by others using different techniques. The methodology and tools have also coped with dialogues from 5 to 31 pages (in graphical form) that are representative of the kinds of dialogues likely to be required for the chosen application area.

6. Conclusions

This section summarises the work in the paper. The results are evaluated, and pointers to future work are given.

6.1. Summary

The goal of this work was to improve on the GUIDE approach for creating spoken dialogues that help people with cognitive impairment to perform daily tasks. The objectives (section 1.1) were as follows:

Simplified Dialogue Design: The first aim was to simplify the design of prompting dialogues. It is believed that this has been successful in that dialogues are now represented graphically. This makes the flow in dialogues much clearer than in GUIDE, and it is easier to modify dialogues.

Automated Dialogue Analysis: The second aim was to automate the checking of dialogues for syntactic and semantic errors. This has also been achieved through automatic translation into LOTOS specifications. Although validation and verification are fully automated, there is still some manual effort required.

For validation, the designer must be prepared to formulate test scenarios using MUSTARD. As it happens, the style of these tests is very similar to what the designer should do anyway and so is likely to require only a little extra work. MUSTARD is aimed at non-technical users, and hides all the details of the underlying specification language, validation technique and tools.

For verification, the designer may be required to formulate dialogue properties using CLOVE. Basic checks such as livelock freedom, guaranteed termination and correct initial behaviour are automated and need very little effort to perform. Only if dialogue-specific properties are required is additional work needed. Even for these, the effort is comparable to that needed for creating test scenarios. CLOVE also hides the technical details of the underlying specification language, verification technique and tools.

Automated Dialogue Implementation: The third aim was to automate the translation of dialogue designs into a form that allows their ready deployment. This has been fully achieved through automatic translation and deployment of dialogue diagrams into a VoiceXML platform.

The approach has been piloted using four significant dialogues for supporting cognitive impaired people: blood sugar testing, hand washing, donning an artificial limb, and making a strawberry smoothie. Despite their size and complexity, these were all successfully described, specified, validated, verified and implemented using CRESS.

The methodology is generic in that it can be used for rigorous design in many application areas. Currently CRESS is used for dialogue services, interactive voice response services, telephony services, web services, grid services and device services.

6.2. Future Work

Several new activities are planned in response to the limitations noted in section 5.3:

- A special-purpose version of the CHIVE diagram editor will be created to make it convenient for designing prompting dialogues. Particular attention will be paid to making it usable by therapists and the like.
- Currently CRESS analyses only dialogue flows. Since the dialogue content is fully defined, it is planned to extend the formal analysis with stylistic analysis. For example, GUIDE has established several good practices for dialogues to support people with cognitive impairment (e.g. those described in Section 1.3.1). Style checks based on linguistic analysis will be added in future using third-party style analysers to check the comprehensibility of dialogue elements.
- A test generation strategy based on [51] will be investigated to automate validation more fully. A graphical test notation will also be considered to make validation more suitable for non-technical designers. As experience with prompting dialogues grows, it is anticipated that checking other desirable properties will also be automated. This will extend the range of formal checks that non-specialists will be able to undertake.
- CRESS dialogues conform to VoiceXML principles. In particular, this requires explicit use of Clear where there are loops in a dialogue. CRESS will be extended to infer such actions automatically, thus simplifying dialogue design. Although a wider range of speech responses will be considered, this aspect will be cautiously developed to ensure that dialogues remain comprehensible to the intended users.
- The Nuance VoiceXML tools used in this work were old versions. Some anomalies were found in their handling of VoiceXML (e.g. audio not being output after invocation of an event handler). The speech tools will be updated, especially if suitable Automatic Speech Recognition and Text To Speech packages can be found (such as those from CereProc).
- Larger-scale evaluations will be carried out, targeting care professionals as the most likely dialogue designers. It will be determined how effectively such designers can create and analyse prompting dialogues. This work will be carried in conjunction with the Brain Injury Rehabilitation Trust in the UK.
- The toolset will be packaged for convenient and easy installation. In fact the toolset is very portable (being written in Java and Perl), so it can run on many platforms including Microsoft Windows, Apple MacOS and other Unix versions. Except for formal verification, the tools do not require a high-performance system. It is therefore planned to create a platform-neutral distribution that can be

installed within minimum technical knowledge. This will allow readier use in end-user homes and in clinics.

Acknowledgements

The work on GUIDE was supported by grant CZH/4/598 from the Chief Scientist's Office, Scotland. Lynne McMichael was supported by the Student Awards Agency for Scotland. Jamie S. Boyd (University of Stirling) undertook the initial development of CHIVE, while Koon Leai Larry Tan (University of Stirling) undertook the initial development of CLOVE. The authors are grateful to Nuance Corporation and the CADP team for the software licences that made this work possible. Frédéric Lang (VASY) provided advice on verification aspects.

References

[1] J. D. Adelinis and L. P. Hagopian. The use of symmetrical 'do' and 'don't' requests to interrupt ongoing activities. *Applied Behavior Analysis*, 32(4):519–524, Oct. 1999.

[2] M. W. Bewernitz, W. C. Mann, P. Dasler, and P. Belchior. Feasibility of machine-based prompting to assist persons with dementia. *Assistive Technology*, 21(4):196–207, Oct. 2009.

[3] B. W. Boehm. Verifying and validating software requirements and design specification. *IEEE Transactions on Software Engineering*, 1(1):75–88, Jan. 1984.

[4] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks*, 14(1):25–59, Jan. 1988.

[5] J. Bongaarts. Population aging and the rising cost of public pensions. *Population and Development Review*, 30(1):1–23, Mar. 2004.

[6] J. Bradfield and C. Stirling. Modal mu-calculi. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*. Elsevier Science Publishers, Amsterdam, Netherlands, 2007.

[7] G. E. Burnett, D. Smith, and A. J. May. Supporting the navigation task: Characteristics of good landmarks. In M. Hanson, editor, *Contemporary Ergonomics 2001*, pages 441–446. Taylor and Francis, London, 2001.

[8] D. Caplan and G. S. Waters. Verbal working memory and sentence comprehension. *Behavioral and Brain Sciences*, 22(1):77–94, Feb. 1999.

[9] N. Caprani, J. Greaney, and N. Porter. A review of memory aid devices for an ageing population. *PsychNology*, 4(3):205–243, Feb. 2006.

[10] S. Carmien. End user programming and context responsiveness in handheld prompting systems for persons with cognitive disabilities and caregivers. In G. C. van der Veer and C. Gale, editors, *Proc. Conf. on Human Factors in Computing Systems*, pages 1252–1255. ACM Press, New York, USA, Apr. 2005.

[11] J. I. M. Carpendale and C. Lewis. Constructing an understanding of mind: The development of children's social understanding within social interaction. *Behavioral and Brain Sciences*, 27(01):79–96, Feb. 2004.

[12] M. Corley, L. J. MacGregor, and D. I. Donaldson. It's the way that you, er, say it: Hesitations in speech affect language comprehension. *Cognition*, 105(3):658–668, Dec. 2007.

[13] Counsel and Care. A charter for change. Counsel and Care, London, UK, Jan. 2008.

[14] P. Ernberg, T. Hovander, and F. Montfort. Specification and implementation of an ISDN telephone system using LOTOS. In M. Diaz and R. Groz, editors, *Proc. Formal Description Techniques V*, pages 171–186. North-Holland, Amsterdam, Netherlands, Oct. 1992.

[15] J. J. Evans. Rehabilitation of executive deficits. In B. A. Wilson, editor, *Neuropsychological Rehabilitation*, pages 53–70. Swets and Zeitlinger, Lisse, Netherlands, 2003.

[16] M. E. Faust, D. A. Balota, J. M. Duchek, M. A. Gernsbacher, and S. Smith. Inhibitory control during sentence comprehension in individuals with dementia of the Alzheimer type*1, *2, *3. *Brain and Language*, 57(2):225–253, Apr. 1997.

[17] Foundation for People with Learning Disabilities. Statistics about people with learning disabilities. *http://www.learningdisabilities.org.uk/information/learning-disabilities-statistics*, Oct. 2010.

[18] H. Garavel, F. Lang, and R. Mateescu. An overview of CADP 2001. *European Association for Software Science and Technology Newsletter*, 4:13–24, Aug. 2002.

[19] T. Gentry, J. Wallace, C. Kvarfordt, and K. B. Lynch. Personal digital assistants as cognitive aids for individuals with severe traumatic brain injury: A community based trial. *Brain Injury*, 22(1):19–24, Jan. 2008.

[20] A. Gillespie, J. Murphy, and M. Place. Divergences of perspective between people with aphasia and their family caregivers. *Aphasiology*, 24(12):1559–1575, Dec. 2010.

[21] L. N. Gitlin, L. Winter, M. P. Dennis, M. Corcoran, S. Schinfeld, and W. W. Hauck. Strategies used by families to simplify tasks for individuals with Alzheimer's disease and related disorders: Psychometric analysis of the Task Management Strategy Index (TMSI). *The Gerontologist*, 42(1):61–69, Feb. 2002.

[22] Headway. Brain injury facts and figures. *http://www.headway.org.uk/facts.aspx*, Oct. 2010.

[23] E. A. Inglis, A. Szymkowiak, P. Gregor, A. F. Newell, N. J. Hine, P. Shadh, B. A. Wilson, and J. J. Evans. Issues surrounding the user centred development of a new interactive memory aid. *Universal Access in the Information Society*, 2(3):226–234, July 2003.

[24] ISO/IEC. *Information Processing Systems – Open Systems Interconnection –* LOTOS *– A Formal Description Technique based on the Temporal Ordering of Observational Behaviour*. ISO/IEC 8807. International Organization for Standardization, Geneva, Switzerland, 1989.

[25] J. F. Kelley. An iterative design methodology for user-friendly natural language office information applications. *ACM Trans. on Office Information Systems*, 2(1):26–41, Jan. 1984.

[26] M. P. LaPlante, C. Harrington, and T. Kang. Estimating paid and unpaid hours of personal assistance services in activities of daily living provided to adults living at home. *Health Services Research*, 37(2):397–415, Apr. 2002.

[27] R. Levison. PEAT: The Planning and Execution Assistant and Trainer. *Head Trauma Rehabilitation*, 12(2):769–755, Apr. 1997.

[28] E. F. LoPresti, A. Mihailidis, and N. Kirsch. Assistive technology for cognitive rehabilitation: State of the art. *Neuropsychological Rehabilitation*, 14(1):5–39, Mar. 2004.

[29] R. Luengo-Fernández, J. Leal, and A. Gray. Dementia 2010: The economic burden of dementia and associated research funding in the United Kingdom. Alzheimer's Society, London, UK, Mar. 2010.

[30] K. A. Lyman. Bringing the social back in: A critique of the biomedicalization of dementia. *Gerontologist*, 29(5):597–605, Sept. 1989.

[31] R. Mateescu and M. Sighireanu. Efficient on-the-fly model-checking for regular alternation-free mu-calculus. *Science of Computer Programming*, 46(3):255–281, Mar. 2003.

[32] C. McCreadie and A. Tinker. The acceptability of assistive technology to older people. *Ageing and Society*, 25(1):91–110, Jan. 2005.

[33] A. Mihailidis, J. C. Barbenel, and G. Fernie. The efficacy of an intelligent cognitive orthosis to facilitate handwashing by persons with moderate to severe dementia. *Neuropsychological Rehabilitation*, 14(1):135–171, Jan. 2004.

[34] A. Mihailidis, J. N. Boger, T. Craig, and J. Hoey. The COACH prompting system to assist older adults with dementia through hand washing: An efficacy study. *BMC Geriatrics*, 8:28, Nov. 2008.

[35] B. O'Neill and A. Gillespie. Simulating naturalistic instruction: The case for a voice-mediated interface for assistive technology for cognition. *Assistive Technologies*, 2(2):22–31, June 2008.

[36] B. O'Neill, K. Moran, and A. Gillespie. Scaffolding rehabilitation behaviour using a voice-mediated assistive technology for cognition. *Neuropsychological Rehabilitation*, 18(1):1–19, Jan. 2010.

[37] S. Pavón Gomez, D. Larrabeiti, and G. Rabay Filho. LOLA user manual (version 3R6). Technical report, Department of Telematic Systems Engineering, Polytechnic University of Madrid, Spain, Feb. 1995.

[38] R. D. Pea. The social and technological dimensions of scaffolding and related theoretical concepts for learning, education and human activity. *Behavioral and Brain Sciences*, 13(3):423–451, July 2004.

[39] C. Pecheur. Advanced modelling and verification techniques applied to a cluster file system. In *Proc. 14th Int. Conf. on Automated Software Engineering*, pages 119–126. IEEE Computer Society, Los Alamitos, California, USA, Oct. 1999.

[40] G. Pijnenborg, F. K. Withaar, J. J. Evans, R. J. van den Bosch, and W. H. Brouwer. SMS text messages as a prosthetic aid in the cognitive rehabilitation of schizophrenia. *Rehabilitation Psychology*, 52(2):236–240, May 2007.

[41] I. M. Proot, H. F. J. M. Crebolder, H. J. Abu-Saad, T. H. G. M. Macor, and R. H. J. Ter Meulen. Facilitating and constraining factors on autonomy: The views of stroke patients on admission into nursing homes. *Clinical Nursing Research*, 9(4):460–478, Nov. 2000.

[42] J. C. Rogers, M. B. Holm, L. D. Burgio, E. Granieri, C. Hsu, J. M. Hardin, and B. J. McDowell. Improving morning care routines of nursing home residents with dementia. *American Geriatrics Society*, 47(9):1049–1057, Sept. 1999.

[43] C. Sharma and J. Kunins. *VoiceXML: Strategies and Techniques for Effective Voice Application Development with VoiceXML 2.0*. John Wiley and Sons, Chichester, UK, 2002.

[44] J. A. Small, G. Gutman, S. Makela, and B. Hillhouse. Effectiveness of communication strategies used by caregivers of persons with Alzheimer's disease during activities of daily living. *Speech, Language, and Hearing Research*, 46(2):353–367, Apr. 2003.

[45] M. M. Sohlberg, S. Fickas, P.-F. Hung, and A. Fortier. A comparison of four prompt modes for route finding for community travellers with severe cognitive impairments. *Brain Injury*, 21(5):531–538, May 2007.

[46] C. A. Stone. The metaphor of scaffolding: Its utility for the field of learning disabilities. *Learning Disabilities*, 31(4):344–364, July 1998.

[47] K. J. Turner. Formalising the Chisel feature notation. In M. H. Calder and E. H. Magill, editors, *Proc. 6th Feature Interactions in Telecommunications and Software Systems*, pages 241–256. IOS Press, Amsterdam, Netherlands, May 2000.

[48] K. J. Turner. Modelling SIP services using CRESS. In D. A. Peled and M. Y. Vardi, editors, *Proc. Formal Techniques for Networked and Distributed Systems (FORTE XV)*, number 2529 in Lecture Notes in Computer Science, pages 162–177. Springer, Berlin, Germany, Nov. 2002.

[49] K. J. Turner. Analysing interactive voice services. *Computer Networks*, 45(5):665–685, Aug. 2004.

[50] K. J. Turner. Formalising web services. In F. Wang, editor, *Proc. Formal Techniques for Networked and Distributed Systems (FORTE XVIII)*, number 3731 in Lecture Notes in Computer Science, pages 473–488. Springer, Berlin, Germany, Oct. 2005.

[51] K. J. Turner. Test generation for radiotherapy accelerators. *Software Tools for Technology Transfer*, 7(4):361–375, Aug. 2005.

[52] K. J. Turner. Validating feature-based specifications. *Software Practice and Experience*, 36(10):999–1027, Aug. 2006.

[53] K. J. Turner. Abstraction and analysis of clinical guidance trees. *Biomedical Informatics*, 42(2):237–250, Apr. 2009.

[54] K. J. Turner. Device services for the home. In K. Drira, A. H. Kacem, and M. Jmaiel, editors, *Proc. 10th Int. Conf. on New Technologies for Distributed Systems*, pages 41–48. Institution of Electrical and Electronic Engineers Press, New York, USA, May 2010.

[55] K. J. Turner and Ji He. Formally-based design evaluation. In T. Margaria and T. F. Melham, editors, *Proc. 11th Conf. on Correct Hardware Design and Verification Methods*, number 2144 in Lecture Notes in Computer Science, pages 104–109. Springer, Berlin, Germany, Sept. 2001.

[56] K. J. Turner and M. Sighireanu. (E-)LOTOS: (Enhanced) language of temporal ordering specification. In H. Habrias and M. Frappier, editors, *Software Specification Methods*, chapter 10, pages 165–190. Springer-Verlag, Godalming, UK, Jan. 2001.

[57] K. J. Turner and K. L. L. Tan. Graphical composition of grid services. In D. Buchs and N. Guelfi, editors, *Rapid Introduction of Software Engineering Techniques*, number 4401 in Lecture Notes in Computer Science, pages 1–17. Springer, Berlin, Germany, May 2007.

[58] K. J. Turner and K. L. L. Tan. A rigorous methodology for composing services. In M. Alpuente, B. Cook, and C. Joubert, editors, *Proc. Formal Methods for Industrial Critical Systems 14*, number 5825 in Lecture Notes in Computer Science, pages 165–180. Springer, Berlin, Germany, Nov. 2009.

[59] UK National Statistics. Population ageing. *http://www.statistics.gov.uk/cci/nugget.asp?id=949*, Oct. 2010.

[60] J. Valsiner. Construction of the zone of proximal development in adult-child joint action: The socialization of meals. *New Directions for Child and Adolescent Development*, 23:65–76, Mar. 1984.

[61] VoiceXML Forum. *Voice eXtensible Markup Language*. VoiceXML Version 2.1. VoiceXML Forum, Piscataway, New Jersey, USA, May 2010.

[62] G. S. Waters, E. Rochon, and D. Caplan. Task demands and sentence comprehension in patients with dementia of the Alzheimer's type. *Brain and Language*, 62(3):361–397, May 1998.

[63] B. A. Wilson, H. Emslie, K. Quirk, and J. Evans. Reducing everyday memory and planning by means of a paging system: A randomized control crossover study. *Neurology Neurosurgery and Psychiatry*, 70(4):477–482, Apr. 2001.

[64] R. Wilson, E. Rochon, A. Mihailidis, C. Leonard, M. Lim, and A. Cole. Examining effective communication strategies used by formal caregivers when interacting with Alzheimer's disease residents during an activity of daily living (ADL). *Brain and Language*, 103(1-2):199–200, Oct. 2007.

[65] A. Wimo and M. Prince. World Alzheimer report: The global economic impact of dementia. Alzheimer's Disease International, London, UK, Sept. 2010.

[66] E. Winner and H. Gardner. The comprehension of metaphor in brain-damaged patients. *Brain*, 100(4):717–729, Dec. 1977.

[67] M. Ylvisaker. Context-sensitive cognitive rehabilitation after brain injury: Theory and practice. *Brain Impairment*, 4(1):1–16, May 2003.

[68] T. Zittoun, A. Gillespie, F. Cornish, and C. Psaltis. The metaphor of the triangle in theories of human development. *Human Development*, 50(4):208–229, July 2007.

[69] I. M. Zmölnig. Pure Data. *http://www.puredata.info*, Oct. 2010.