

Development of a Generic Multimodal Framework for Handling Error Patterns during Human-Machine Interaction

Gregor McGlaun, Manfred Lang, and Gerhard Rigoll

Institute for Human-Machine Communication
Technical University of Munich
Arcisstr. 16, 80290 Munich, Germany
phone: +49 89 289-28541

{mcglaun | lang | rigoll}@ei.tum.de

ABSTRACT

In this contribution, we present a generic and therefore easily scalable multimodal framework for error robust processing of user interactions in various domains. The system provides a generic kernel for evaluating user inputs and additional pieces of information from situational, personal, and functional context. After an initial domain-specific configuration, the system is capable of detecting a set of error situations and patterns. In case an error is likely to occur or detected, a context-adequate dialog output is generated. For classification of the error patterns and the selection of the according dialog strategy, we have implemented a fuzzy-logic algorithm, using Mamdani controllers. The multimodal framework has been applied and evaluated in two application domains: an in-car infotainment and communication system and a 3D virtual shopping mall in a desktop PC environment. From a large user test, we have transcribed eleven error scenario contexts each consisting of 15 individual test sets, and analyzed them in an offline evaluation. In the VR domain, the rates for a correctly detected error pattern have been between 90.7% and 95.0% (86.7% up to 94.3% in the car domain). The rates for the appropriately selected error resolution strategy have been between 93.9% and 96.3% (91.0% up to 96.1% in the car domain).

Keywords: Generic, multimodal, system, error, handling, fuzzy logic, human-machine interaction, automotive, VR;

1. INTRODUCTION

In modern applications, an enormous increment of various technologies can be realized. Due to the price decline of many electronic devices, their compact size, and their great capability, today's technical systems offer a large spectrum of functionality. Yet, as a direct consequence of the functional complexity, the interaction with such interfaces is getting more and more difficult for the user

which often leads to different kinds of operation errors. The error patterns are highly domain-specific as well as the factors leading to them. This paper exemplarily focuses on two types of application systems in completely different domains: *infotainment and communication systems established within a car* and a *3D virtual shopping mall in a desktop PC environment*.

Especially in the car domain, often error-prone situations occur regarding the human-machine interaction with different in-car applications, as the driver often has a certain mental workload. This basic stress level is due to the execution of so-called *primary* (navigation, stabilization of the car, etc.) and *secondary tasks* (windshield wiping, honking, etc.), and may be increased by environmental impacts, like the conversation with a co-driver. If the driver interacts, e.g., with a communication and infotainment system in such a stress phase (*tertiary task*), inattention, distraction, and irritation occur as a consequence of the high workload resulting from a superposition of the tasks mentioned above, which will become manifest in an increased error potential and in erroneous operations of these tertiary systems.

Compared to the automobile domain, in front of a desktop PC, the user can predominantly execute her or his operations in a concentrated way, as there is no dual task competition. Error patterns are assumed to be less influenced by the situational context induced by the environment. Hence, typical errors caused by the user mainly occur in three action categories: *navigation*, *picking*, and *manipulation*.

In both domains, also errors are likely to occur which are caused by a certain malfunction of the technical system. For example, many speech interfaces are still noise-prone, interfaces for gestural input have the disadvantage of being sensitive to changes in lighting, and, if worn out, even a button can have a malfunction. If the system has a monomodal interface, the error aggravates, as the system can no longer be properly used until the cause is found and removed. Therefore, in the many domains, increasingly *multimodal interfaces* (MIs) are applied. Besides

classical *tactile* sensors (buttons, turning knobs, etc.), MIs also allow for *speech*, *gestures*, or both as interaction paradigms[1]. Compared to monomodal systems, MIs provide significantly shortened learning periods as well as a highly individual and intuitive form of interaction, since they meet natural human communication habits[2]. If, for any reason, the chosen modality channel is disrupted, the user can be suggested an alternative input form as a fallback in dependence of the given functionality and the situational context. The following exemplary scenario might appear in the car domain: caused by heavy environmental noise, the recognition performance of a speech recognizer drops drastically. The system could now alternatively suggest the driver tactile input, if the car is just waiting at a traffic lights. Oviatt et al. mention that in dedicated scenarios, it is possible to omit up to 86% of all task-critical errors, if only an alternative input modality is present[2]. When the user provides input in a *synergistic* and *redundant* way, mutual ambiguities can be eliminated[2]. To distract the user as less as possible, and, on the other hand, to increase operation comfort, a context sensitive error management is imperative. The goal is to duly identify potential error sources concerning the operation of the applications mentioned above, and to trigger according measures, e.g., warning outputs (*a priori-error management*). If errors have occurred, they should be solved rapidly, effectively, and transparently for the user to prevent further distraction and error propagation (*a posteriori- error management*).

2. PRELIMINARY WORK

The problem of error robustness during interaction with interfaces in the automobile has been discussed in numerous preliminary works. In an essay, D. Norman criticizes that the automobile industry often ignores the advances in user-interface design. In his opinion, many existing interfaces have poor ergonomics, tiny readouts, or way too many buttons so that a purely tactile operation of such systems is very unsafe while driving[3]. A research team at the Institute for Technical Computer Science (RWTH Aachen, Germany)[4] deals, amongst other things, with the intuitive design of multimodal on-board car systems. A special focus is placed on the individual adaptation to the user in different operation situations.

In the VR domain, C. Wewerka et al. show a concept of a component-based virtual mall implemented in Java 3D[5]. The architecture allows for easy integration of arbitrary input modalities. For an error-robust and intuitive way of interaction, the selection paradigm contains an adaptive picking ray that can be given the form of a cone. In his work, M. Turunen studies different error types that occur during human-machine interaction with a speech-based interface in the context of virtual worlds, and introduces some improved error-handling methods[6].

3. THEORETICAL BACKGROUND

Generally, we define an error in human-machine interaction, if the user does not reach her or his desired goal, and no coincidence can be made responsible for it. For a systematic classification of error types, we basically assume that either the user or the system can cause an error in the human-machine communication process. Consequently, two main error categories, which are characterized below, can be distinguished: *user-specific errors* and *system-specific errors*. This section closes with a formal discussion of methods and strategies for error resolution.

User-Specific Errors

The basic taxonomy underlying our modeling of user-specific error classes is related to the work of J. Rasmussen and J. Reason[7]. Reason identifies three different levels for classifying errors. Analyzing the reason of the error, he differentiates between *formal* attributes of the error (“what“?), the *circumstances* under which the error occurred (“where“?), and the *cognitive mechanisms* involved („how“?). According to the level model of Rasmussen, three basic error levels can be identified.

The *skill-based level* (SBL) comprises smooth, automated, and highly integrated routine actions that take place without conscious attention or control. Human performance is governed by stored patterns of pre-programmed instructions represented as analog structures in a time-space domain. Errors at this level are related to the intrinsic variability of force, space, or time coordination. Sporadically, the user checks, if the action initiated by her or him runs as planned, and if the plan for reaching the focused goal is still adequate. Error patterns on SBL are execution or memory errors that result from inattention or overattention of the user.

Concerning errors on the *rule-based level* (RBL), the user violates stored prioritized rules (so-called productions). Errors are typically associated with the misclassification of situations leading to the application of the wrong rule or with the incorrect recall of procedures.

At the *knowledge-based level* (KBL), the user applies stored knowledge and analytical processes in novel situations, in which actions must be planned on-line. Errors at this level arise from resource limitations (bounded rationality) and incomplete or incorrect knowledge.

System-Specific Errors

In our error taxonomy, we also address errors caused by the system. Well-known examples are recognition error, like misinterpretation, false recognition of a correct user input, or an incorrect system-intrinsic activation of a speech recognizer (e.g., the user coincidentally applies the keyword which activates the speech recognizer in a conversation with the co-driver). Moreover, processing errors (timing problems or contradictory recognition results of different monomodal recognizers, etc.) as well as technical errors (e.g., system overflow or breakdown of

system components) can occur. The superposition of errors represents a more complex error pattern. E.g., a speech command of the user, which is not allowed in a certain mode, is mapped onto a valid command because of an erroneous interpretation of the system, but on the other hand, this command does not correspond to the intention of the user. Other error classes, which, however, are not considered in the final implementation, are output errors (e.g., insufficient or incomprehensible information) and global design errors, like bad ergonomics.

Taxonomy of error resolution strategies

If potential or definitive system- or user-specific errors have been identified, the system tries to solve them via a dedicated system dialog. Feasible strategies (warning, request for reentry, request for changing the input modality, selection from command alternatives, tutorial, forcing functions, gagging, self-correction, etc.) can be distinguished by some criteria, as follows: initiation of the error notification, context-dependency, consideration of individual characteristics of the user (e.g., her or his state of knowledge and awareness), impact and demands of the strategy (i.e., level of detail of the knowledge transfer and the extent of the intervention), as well as the inclusion of the user (i.e., the degree of interactivity during the error resolution process and the form of the system feedback). The selection of the dialog strategy substantially depends on contextual parameters that, amongst others, are the personal state of the user (e.g., emotional behavior), environmental data (e.g., background noise, lighting conditions), as well as the system context (e.g., currently chosen input modality, state and mode of the application).

4. SYSTEM SPECIFICATION

In this section, we describe the system architecture and characterize the functionality of the single components of the multimodal error-robust framework.

Framework Topology

The system architecture basically consists of three main processing levels: the *input level*, the *integration level*, and the *output level* (see figure 1).

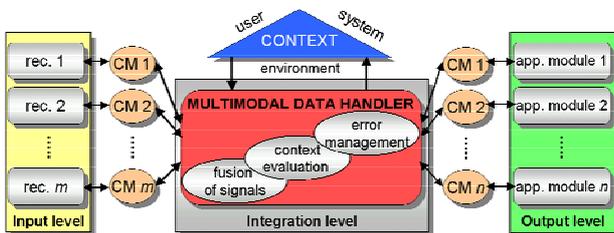


Figure 1: Multimodal framework based on a late semantic fusion of input; the error management module is part of the generic core at the integration level

The input level contains any kind of interface that is capable of recognizing user inputs (e.g., mouse, buttons, speech recognizer, etc.). Dedicated command mappers (CMs) encode the information bits of the single independent modality recognizers and context sensors into a meta language based on a context-free grammar (CFG). In the integration level, the recognizer outputs and additional information of context sensors (e.g., information about application environment, user state, etc.) are combined in a *late semantic fusion* process that is extensively illustrated in [8]. The error management component, that will be subsequently described, is also located in the integration level. Unless the system realizes any potential or type of error, another set of CMs translates back the resulting command modeling the user intention into an application-specific command. By this principle, all core components of the framework, that are located at the integration level, can be kept generic and independent from the specifications of the individual recognizers and the application components to the greatest possible extent. A configuration script must only once domain-specifically adapt the modules for signal fusion, context evaluation, and error management.

In the following paragraphs, we will describe the structure of the error management module and the functionality of its single components. The error management process consists of four steps: *error feature extraction*, *error analysis*, *error classification*, and *error resolution*.

Error Feature Extraction

This component of the error management module continuously extracts certain features from the stream of incoming messages. The relevance and structure of these features has been determined in a preliminary test series [9]. Generally, we distinguish between *command features* (competing actions, correcting actions, redundant interactions, command repetitions, modality changes, time relations between commands), and *contextual features*, such as user characteristics (system experience, domain-specific knowledge, emotional features, etc.), environmental influences (e.g., background noise, lighting conditions), and system-intrinsic parameters (e.g., output volume, confidence measures of the individual recognizers).

Error Analysis

The verification of an error potential and the detection of an error pattern are managed via *Error Analysis Sentinels* (EAS). EAS are highly specialized dynamic agents with each EAS designed for the verification of an individual error pattern. A single sentinel can be regarded as an instance of a class of generic sentinels. Thus, the architecture is easily portable to other domains and freely scalable in its extent. For the validation of an individual error pattern, we use a *fuzzy-system* approach. The method was chosen, as it needs very small training, and represents a fast realization base in terms of a proof of concept. Due

to the fuzziness of the formulated productions, the approach is clearly more flexible compared to a classical expert system with hard decisions. The EAS were implemented as Mamdani controllers. Depending on the EAS, the rule corpora comprise 15 to 103 rules with one rule containing between three and eight input variables (components of the feature vector). For each input variable, there are two to seven fuzzy-rule blocks with the membership functions for the single fuzzy sets implemented as triangular functions. The defuzzification is done using the center of gravity (COG) method, as in general, it has a smooth control behavior. Each EAS evaluates the respective features on the basis of its individual criteria catalog, and calculates a probability as well as a time stamp for the represented error type.

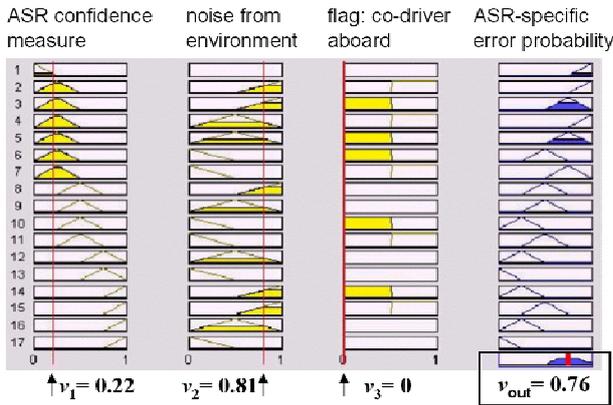


Figure 2: Example for computation of the probability of a speech recognition error in an automotive scenario

In figure 2, a simplified example is visualized to show the principle of the algorithm. Hence, let the output variable v_{out} be the system-specific error for an automatic speech recognizer (ASR) that is applied in the car domain. The fuzzy rule corpus totally comprises 17 individual rules r_i , $i \in \{1, 2, \dots, 17\}$, each of them featuring three input variables (all values normalized): the confidence measure of the automatic speech recognizer (v_1), the volume of environmental noise (v_2), and the flag ‘co-driver aboard’ (v_3). v_1 has four fuzzy rule blocks, v_2 a total of three, and v_3 two rule blocks. Assumed the EAS takes values $v_1 = 0.22$, $v_2 = 0.81$, and $v_3 = 0$, the rules r_1 , r_3 , and r_5 will apply. By superposition of the respective membership functions of the output variable, we get $v_{out} = 0.76$ which can be interpreted as the ASR-specific error probability.

Error Classification

As a result of the error analysis, each EAS delivers a probability for its individual error type. In this phase of the error management process, the resulting error type(s) are determined. For this decision process, dynamic threshold logic is applied. All error types with a probability over the threshold are fed into the error resolution module as input variables. Thus, the system is even capa-

ble to recognize superposed errors and error propagation patterns (as regards content and point of time). However, in the current state of development, the management of overlapping errors is not yet implemented and will be part of further work.

Error Resolution

For the selection of a dedicated dialog strategy, the current context parameters as well as the error types are fuzzified as singletons in rule sets. There is a total of twelve different correction strategies (see section 3). Each strategy type is instantiated in terms of an error strategy sentinel (ESS). In direct analogy to an EAS, on the basis of the input variables, each ESS determines a plausibility measure for its represented strategy, using a Mamdani controller. From the results, the strategy with the highest plausibility is chosen. The strategy template is varied by additional parameters with regard to form (e.g., choice of words, intonation, dialog length) and content (e.g., providing a suitable fallback modality). For an individual assignment of each parameter, a Sugeno controller is used. The symbol collector gathers the parameter values, customizes the selected template, and generates the respective action and the dialog output. A detailed description of this adaptation algorithm will be presented elsewhere.

5. SYSTEM EVALUATION

Both in the automotive and the VR domain, we have collected test data from 40 subjects in 25 potential error scenarios. A part of the data has been used for the domain-specific configuration, the deduction of the rules, and the training of the rule base. The other part has served as an input data set for the evaluation of the system.

Test description

The automotive trial platform was a test car in a laboratory environment. Test persons had to follow a predefined street course in a laboratory driving simulation. Additionally, they had to perform diverse tasks regarding the operation of the multimodal in-car system. The target application was an infotainment and communication system, consisting of an MP3-player, radio, telephone, and a WAP simulation. As tactile modalities, the user was provided a 10” touch screen located at the center console, as well as an eight-button array integrated in the armrest. Input could also be delivered via natural speech (30 different commands with keyword-initializing). Moreover, the subjects could use a set of 15 dynamic hand gestures. For interaction, subjects had to move their right hand into the focus of the camera. To initialize the gesture recognizer, the subjects had to shortly splay their fingers before putting in the desired gesture.

The other test platform was a desktop PC environment. As a target application, we have used a virtual shopping

mall implemented in VRML. Test subjects could interact with the scene by navigating in different department stores. They were given manipulation and picking tasks (e.g., selecting items, and putting them into their shopping cart. As tactile input modalities, the subjects could interact via a keyboard, a conventional mouse, a 6-DOF mouse, and a joystick. The scene was displayed on a 19" touch screen. The complexity of the vocabulary was comparable to that in the automotive test environment.

In the first part of the test, we focused on user-specific errors. Thus for a targeted provocation of these scenarios, the test supervisor has interpreted all inputs except for tactile modalities (partial wizard-of-oz trial). In the second part of the test, real recognizers were used. The aim was to collect data of system-specific errors. All interactions of the user and all measured context parameters (e.g., data from the driving simulation in the car domain) have been logged. When the test series had been finished, the constellation within the error scenarios has been transcribed and filed in terms of test data sets.

Results

We have trained the developed error management system in eleven error scenario contexts (5 SBL, 3 RBL, and 3 KBL) with 20 sets of training data. In an off-line evaluation, the system has been analyzed in these scenarios with 15 test data sets each. Training and test data sets have been strictly disjointed. First we evaluated the power of discrimination between user-specific and system-specific errors (see table 1).

		car domain		VR domain	
		USE	SSE	USE	SSE
classified	actual				
	USE	95.3%	4.7%	97.8%	2.2%
SSE		3.3%	96.7%	1.5%	98.5%

Table 1: Domain-specific confusion matrix (USE = user-specific errors, SSE = system-specific errors)

In the car domain, a significantly higher set of misinterpretations occurred which might be due to the dual task problem (see section 1). This explains why the confusion values are slightly higher than in the VR domain. Generally, the system had the tendency to rather interpret a system-specific error as user-specific than vice versa.

		car domain		VR domain	
		ECR	SSR	ECR	SSR
error type	rate				
	SBL	94.3%	96.1%	95.0%	93.9%
RBL		90.3%	92.7%	94.3%	96.3%
KBL		86.7%	91.0%	90.7%	96.1%

Table 2: Domain-specific rates for correctly classified error types (ECR) and correctly selected resolution strategies (SSR) in dependence of the skill-based (SBL), the rule-based (RBL), and the knowledge-based level.

The overall rates for a correctly detected error pattern and the appropriately chosen strategy are displayed in table 2. In comparison to SBL, error patterns at KBL are often very complex, which results in worse recognition rates. This holds for both the car and the VR domain.

5. CONCLUSIONS AND OUTLOOK

In this contribution, we presented a generic multimodal framework that was implemented in two different application domains. The evaluation results show the applicability of the fuzzy algorithm as a proof of concept.

Further work in this research field will comprise a field test with a user-centered acceptance evaluation of the system in which also real recognizers for speech and gesture recognition will be applied. In an ongoing project, we are implementing an algorithm for coping with temporal error superposition and error resonance scenarios. Moreover, we plan to evaluate the qualification of statistical and hybrid classifiers (Hidden-Markov Models and Neuro-Fuzzy Systems) for discriminating error patterns.

6. REFERENCES

- [1] Project FERMUS (Error-Robust Multimodal Speech Dialogs), in: *www.fermus.de*, 2003
- [2] S. Oviatt, et al: "Error Resolution during Multimodal Human-Computer Interaction," ICSLP 1996, Philadelphia, USA, in: *Proc. Vol. I, pp. 204-207*, 1996
- [3] D. A. Norman: "Interaction Design for Automobile Interiors," in: *Popular, Recent Essays, www.jnd.org/dn.mss/InteractDsgnAutos.html*, 2003
- [4] S. Akyol, et al.: "Multimodal Usage of Adaptive Car Board Systems," in: *Kraftfahrzeugfuehrung, Springer, Berlin*, 2001
- [5] C. Wewerka, et al.: "A Virtual Mall in Java 3D", NetObject Days 2000, in: *Electronic Proceedings CD-ROM, Erfurt*, 2000
- [6] M. Turunen: "Error Handling in Speech User Interfaces in the Context of Virtual Worlds," in: *Proceedings of ACHCI'98, Report B-1998-4, pp. 68-75*, 1998
- [7] J. Reason: "Generic Error-Modeling System: A Cognitive Framework for Locating Common Human Error Forms," in: *New Technology and Human Error, chapter 7, pp. 63-83*, 1987
- [8] G. McGlaun, et al.: "A New Approach for Integrating Multimodal Input via Late Semantic Fusion," USEWARE 2002, Darmstadt, Germany; in: *VDI-Bericht 1678; pp. 181-185*, 2002
- [9] F. Althoff, et al.: "Evaluating Multimodal Interaction Patterns in Various Application Scenarios," GW 2003, Genoa, Italy, in: *Post-Proceedings: Lecture Notes in Artificial Intelligence, Springer, Berlin*, 2004