2002-07

Human Factors Based Tools for Dependable Interactive Systems Development

Luqi; Guan, ZhiWei

Monterey, California. Naval Postgraduate School

NPS-SW-02-009

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Human Factors Based Tools for Dependable Interactive Systems Development

Technical Report

By

Luqi, ZhiWei Guan

July 2002

Prepared for: NSF

PROJECT SUMMARY

The objectives of the proposed research are to develop an integrated framework and supporting tools for the development of a dependable interactive system. This proposed research promotes steps toward the integration of human factors within the software activity model by considering human characteristics in the development of interactive systems. Development areas impacted include system specification, software model verification, and system testing simulation. The basic concept of this proposal is to consider the human as part of the system during the analysis of system functionalities and properties. Human inclusion in the entire system, not just software, during verification and testing simulations would improve the dependability of interactive systems.

This proposal framework would build an iterative loop for the development of interactive systems from system specification, to model verification, and to prototype validation.

The proposed human involved system specification would provide a specification method, for human involved system behavior and interactive system behavior attributes. Research will be conducted on the decomposition of goals of human involved software interactive systems. Interlink of interactive sub-goals and the relationship between goals and action will also be addressed. Analysis of the interactive system behavior attribute will provide a description of the overall attributes for system interaction requirements specification.

Following the development of software, software design models will be built and developed based on the description of system requirements and attributes. In this proposed work, the extraction of a software behavior model from whole software design model, describing what, when and how software would perform its operation, will be studied. Model checking will be employed to ensure the software model is consistent with the requirements of system operation. The state explosion problem of model checking will be avoided by extracting the software behavior model and the black-box description of human behavior.

Based on the verified software interactive functions, a prototype system can be built. We will develop a "Wizard of Human" (WoH) interactive system testing simulation technology to test the interactive system. The Wizard Human AutomaTic Fault Injection Testing Simulation (WHAT-FITs) is based on the test cases generated from the system requirement specification. WoH serves as an agent to perform the human action based on the requirement specification. Human error fault injection based on the definition of human behavior and system interaction states, which include safety constraints, function correctness, security constraints, time constraints and quality of service, will be applied to ensure software constraints. The overall satisfaction of software constraints would insure the dependability of the implemented interactive system. A graphic display of counter-example model checking will be illustrated based on the prototype design system. The display of test results and interpretation based on software constraints will be presented. Development feedback will help users fulfill dependable system development.

This proposed work would eliminate the misunderstanding between interaction requirements and software design thereby improving the reliability and dependability of interaction, as the design model will be verified and the software implementation will be validated with the interaction specification of the system. Safety of the system will be improved as human behavior is considered in the interactive system development process reducing software disasters in mission/life critical situations. The total cost of system development will be reduced with the automatic testing of human interaction. Further human behavior analysis of software implementation will be reduced, as the WoH has acted as the human to check the consistency of software design behavior in advance.

This research is covers areas, which include system specification, verification, and testing. The purpose of technology integration is to introduce improvement and to allow the integrated components to complement each other, increasing the overall efficiency as a whole. By integrating these techniques, we can improve the formalism of interactive system development, reduce the cost of human action involved interactive system testing, reduce development and maintenance time, and improve the dependability of interactive software. Additional benefits will include information for human operation training and continuous support for interactive software evolution.

This research could have broad impact on the area of critical aerospace and communication domain, including mission control for manned and unmanned missions, air traffic control, and onboard aerospace vehicles. Consideration of human interaction at the beginning of software development will increase the reliability and safety as well as reduce the risk of software failure.

PROJECT DESCRIPTION

C.1 Objectives and significance

The dependable systems include autonomous system and combination of human operated and automated systems. For interactive system, especially safety critical system, although the advanced automation of the system make much improvement of software dependability, the complexity of the system interaction always makes the confusion of what the automat are doing potentially comes out. The reason for the behavior inconsistency is incomplete and ambiguous description and application of system behavior during the system development. The main objective of the proposal work is to enable the human action to be considered into the system development process to help the clearance of software interaction, so that to improve the system dependability. We proposed to develop a framework and several technologies that are needed to realize the dependable interaction behavior insurance, which include the system interaction specification, software behavior model verification, automatic interaction testing with wizard human operation and human error injection.

The proposed interaction requirement specification tool will enable the explicit description of system behavior in the requirement phase, in which human behavior description is included as part of the system specification. The safety and liveness properties of interaction will be specified. To ensure the software design model to be consistent with what software supposed to be doing, the software behavior model will be abstracted from the system design model and be applied with model checking. The verification of software model will increase software function correctness and reliability.

Furthermore, to make sure the software implementation is performing as it supposed to. Validation of system implementation will be performed. Wizard Human AutomaTic Fault Injection Testing Simulation (WHAT-FITs) technology will test the software operations. Wizard Human would perform the human actions based on the human behavior specification. Human error will be generated based on the human behavior and runtime interface patterns. Fault injection of human error will test the software reliability and safety. The tested software performance will be evaluated according to the software constraints definitions.

Further counter example of model checking and the test result and experience of WHAT-FITs will be display to the end user with easy and useful knowledge to perform the system modification, so that to increase the final software dependability.

C.1.1 Significance and benefits

This technology will enable the following to occur:

1. The entire requirement of interaction behavior of system (human involved software) will be considered at the beginning of software development.
2. The constraints of interaction (safety and liveness properties) will be explicitly specified.
3. Software behavior model can be abstracted from the software design model based on the interaction specification.
4. Verification of software behavior with behavior requirement specification and interaction constraints by using properly selected model checking method.
5. Generation of test scenarios from the interaction behavior specification.
6. Creation of Wizard Human based on the human behavior description in the system interaction specification.
7. Intentional human error generation mechanism for Wizard Human to perform the error injection at the each possible selectable state transition.
8. Integrated WHAT-FITs environment to perform the software interaction testing.
9. Graphic display of the counter-example of model checking and result and scenario interpretation of WHAT-FITs.

C.1.2 Technical barriers

Previews methods for achieving reliable interactive system, including statistical testing ad mathematical proofs of software properties, have the disadvantage of being slow, labor intensive and expensive. Also the absent description of human behavior will not make sure all the designed transitions behaved as it designed to during the real human software operation. What is needed is a theoretical basis and systematic methods for construction formal

development framework for a variety of dependable interactive system. The iterative procedure of interaction formal specification, software design formal model checking, and rule-based fault injection test simulation will make the refinement of human involved software operations, and improve the dependability of the interactive system.

C.2 Technical approach

C.2.1 Formal Specification of Human Involved System Behavior

The proposed work under this task is to improve and extend the current software requirement specification style to allow for more detail description of human involved system behavior requirement. The current software specifications include the properties descriptions of software. For interactive system, the specified properties would be the description of human and software interaction, which can be described as stat transitions based on the certain condition. It can give out a clear description of what the software should do for user, and what software can do based on the trigger of user's action. In the past literature, the software behavior is considered with high confidence. Human's actions are totally ignored. In our consideration, if the supposed system behavior (both supposed software behavior and human behavior) can be described clearly and completely, it can be used as the high-trust criteria for whole software development. Also, further analysis and refinement of the system behavior specification will help to generate reasonable and human nature prone system behavior, which would improve the trustworthy of the system, which is also defined as system's "dependability".

This proposal would provide a specification method, to specify the human involved system behavior. This description approach would be goal-based, rather than state-based approach. Both the constraints of safety and liveness properties of the interaction will be described. Research would be done on the decomposition of goal of software/human-based system, interlink of sub-goal, the relationship of goal and action. Also, the strategies of goal driven interaction, and the sequence of action and state will be studied and described. Totally, the description of goal-based system interaction may include the newly defined elements, such as goal, action, strategy, action element, evaluation criteria. The extending usage of temporal logic will be studied to fit the description requirement.

Beside the description of interaction requirement, we will also explore open specification methods for other properties of interest in distributed environment within the uniformed framework, such as the time constraint of interactive, safety requirement, interlock negotiation strategy.

Our past work has been done in the requirement solicitation by using graphic illustration. Passed successful experience of computer aided requirement specification proved that it can help the clear generation of requirement and easy of usability. The properties described in the current Computer Aided Prototype System include the time constrains and control constraints [52][53]. In this proposal work, we will try go forward to provide a more comprehensive description of interaction and employ a graphic environment for realize the specification of interaction system.

C.2.2 Software Behavior Model Extracting and Verification

Based on the system requirement specification, software design model will be built. There are many aspects will be concerned in the software model, such as aspects of interactive behavior, data control, time sequence, parameterized functionality. In this proposed work, the extraction and abstraction of software behavior model from system specification will be studied, which will be applied with model checking with the original behavior specification. Software behavior model describe what, when and how software should perform the operation with its environment, which of cause include human. The abstraction of software behavior model will be done based on the analysis of the software sub-goal, action sequence, and state transition step by step. Primary considered software behavior model would be consisted by state, pre-condition, status, actions, and post-conditions.

Second, we need to go further to check the abstracted software behavior model to ensure the software design to be correct, precise, and unambiguous. Model checking will be used to check if the software design model is consistent with specified behavior model, for special interactive aspect. The successful application of model checking in hardware system greatly increased the hardware reliability and dependability. For mitigation of model checking

from hardware to software is the state explosive problem. The state of software is increased very fast together with the complexity of software. In this proposal, we supposed to justifiably apply model checking to verify the software interactions. The abstraction of the software behavior model will shrink the source of software state checked to be relatively more precise, and limited. The purpose of checking the software interaction design model is to increase the function availability, reliability, and survivability of software interaction.

For interactive system, the property with high concerned should be studied. First, interactive system may include many information exchange and communication. The process of software and human may be high concurrent. Especially, the synchronous of interaction should be checked at the first priority. Second, the real time properties of interaction would be verified. To identify timeliness properties of interactive system that may in some extend affect the dependability of the interactive system.

C.2.3 Wizard Human AutomaTic Fault Injection Testing Simulation (WHAT-FITs)

With verified software interactive functions, the reliability, safety and survivability of system behavior should be re-concerned based on overall consideration of whole system, which include human, environment and software. In this proposal work we will develop "Wizard Human" Interactive System Testing Simulation technology. The testing simulation is performed based on the test scenario generated from the system requirement specification. To avoid the numbers of test case exploring, the test simulation cases is concerned with critical goal driven situations. Situations with mode change, feedback display, and user explicit actions will be firstly concerned.

The "Wizard Human" serves as a wizard agent, which will act like human. The difference between wizard of human and real human in the test simulation is that action of wizard of human is based on the rules mitigated from human centered system interaction specification. According to the interaction specification, the human action rules can be formalized, which we called "mental rule". Mental rule is the action base for human to performing the interaction. During the testing, the application of mental rule is based on the evaluation of testing context situation. The context situation includes software states and the environment context at the time of interaction. After actions based on mental rule are performed, the software performances will be recorded, which include data structure transferred and the resulted satisfaction of software attributes. Software attributes is to evaluate the system overall properties, which include safety constraints, function correctness, security constraints, time constraints and quality of service.

Also, human error for testing will be generated based on the mental model and the runtime action behavior. The generated human errors will be used to realize all of the possible opposite actions. Human error based fault injection mechanism will greatly improve the efficiency of software testing. Especially, the liveness properties of interaction will be validated.

During the WHAT-FITs process, system is performed following the test scenarios. Once the testing simulation reaches the interface of wizard human and system, the runtime interaction situation will be identified according to the software states. Actions according to the wizard human's mental model will be applied. Human error fault will be injected into the system performance. After the performance of action, the following state of software will be checked with the post-condition of mental rule. Safety concern, function result, time satisfy, and quality of action will be measured and evaluated. The satisfactory of post-condition of interaction means success of a test case simulation. If the check of post-condition is fail, the reason of failure can be identified, which can be retraced along with the test scenario, software state, and related human actions and source errors.

C.2.4 The visualization of result and interpretation of verification and testing

Software behavior model checking will give out the counterexample of software state transmission. Visualization of the counter-example in the software prototype development process will help the designer to perceive the design problem. The visualization of test result will also give the illustration of system performance with human error. The defects of interaction design such as action inconsistent, mode confusion, and data inconsistent will be analyzed and illustrated.

Also, during the test simulation, user and software experience will be monitored, which include the frequency, sequence, and the timing of usage of mental rule. Further analysis of human behavior specification can be performed based on the experience data. Unreasonable behavior design and specification will be analyzed. It will give extremely useful information for the early eliminate of behavior design defects. Also, for those feasible behavior designs, the statistic analysis of behaved actions can give the practically useful suggestions for future training of operators to perform the system interactions.

C.3 General plan of work

C.3.1 Plan overview

We envision the software prototype system that we currently have as the starting point for additional further research. There are several extension will be done according to the characters of the interactive software. The specification of human involved interactive system will be studied. The measurable attribute of safety, reliability and usability of the interactive system, such as safety constraints, function correctness, security constraints, time constraints and quality of service, would be specified. For special, the concurrency and real time of interaction will be identified and described. The measurable attribute of concurrency and real time constrains of interactive system will be studied and specified.

Based on the system requirement specification, the model of the human involved system will be built and refined. The tools for system modeling can employ current computer aided prototype system. The abstraction of software behavior model from whole software model will be studied. Related tools will be built to enhance the current modeling tool. In additional, the tools for software behavior model checking will be built for determining whether systems have desired dependability properties. The model checker will be employ current widely used model checker, selected from SMV, SPIN, FDR, etc. The selection of model checker is based on the study of the suitable for checking of concurrency and real time attributes of interaction. The checkable properties for model checker are the defined measurable attributes in the system requirement.

With the checked model of software, the prototype system can generate software implementation. By providing the testing simulation, dependability property of software can be validated and evaluated. Test scenarios will be generated from the system specification. Wizard Human AutomaTic Fault Injection Testing Simulation environment will be built to test the performance of software implementation by human error injection. Also, as the representative of human, wizard of human will be involved in the system and will act as human agency.

Further more, an additional facility for visualization of checking result and the testing result will be studied. It will provide the user more easy way and useful channel to perceive the defect of software. Graphic display of software model will be built to illustrate the analysis result of testing.

C.3.2 Broad design activities

The steps in creating an integrated development framework of interactive system are as follows:

1) Study the current requirement specification methods and their focalization, which are mostly special purpose designed. Investigate appropriateness of methods for using in the human-software interactive system specification.
2) Study the attributes, such as real time and concurrency, and define the constraints, such as safety, reliability, and availability, for interaction system specification.
3) Modify the current software prototype system for capability of modeling the interactive character in the system.
4) Modify the current software prototype system for capability of illustrating the system attribute for both functional evaluation and nonfunctional evaluation.
5) Study the model abstraction method for software behavior extraction with related attributes of interactive system, and select and apply appropriate modeling checking method for the interactive properties.
6) Extend the capability of prototype system by including the test scenarios generation from the system specification. The scenarios generation method will be studied based on the interactive characters, such as interactive time constrain, concurrency, and so on.
7) Generate human errors for fault injection in the testing based on the human behavior specification and the possibility of action selection designed in the software.

8) Creation of the WHAT-FITs environment by building the wizard human agent, which performed based on the mental rule abstracted from the behavior specification of system with human related aspect.
9) Graphic result illustration of counter-example of model checking and the result of testing simulation in the prototype interactive system.

C.3.3 Deliverables

1) Extended specification method for system interaction behavior description with human actions included.
2) Interaction behavior attributes description for dependability of the interactive system.
3) Software behavior model abstraction technology based on the system interaction attributes.
4) Appropriate model checking method for verification of interactive properties and software behavior model.
5) Test scenarios generation technology from system interaction specification.
6) Generation technology of "Mental rule" of Wizard human from system specification with human aspect.
7) Human errors generation technology for fault injection in the testing.
8) Wizard Human AutomaTic Fault Injection Test Simulation environment.
9) Graphic display of counter-example of model checking and testing scenario in current prototype system.

C.3.4 Description of procedure

Building on our strengths, we will perform the following:

1) Further develop the description language with full consideration of behavior properties of interactive system, in which the description of human behavior will be included.
2) Further develop the description of interaction properties and measurable attributes of interactive system in the software design modeling.
3) Further include the software verification method for model checking of interaction properties.
4) Further include the graphics display of counter-example of interaction model checking.
5) Construct prototype software tools that realize methods enabled by (1)(2)(3)(4).

Also, we will build a test scenario generator, which can automatic generate the test scenario from the system interaction specification. The specification of human behaviors will be extracted from system specification and generate "mental rule", which will serve as the action rules for the wizard human in the testing simulation. Human error generation rules based on the human behavior specification and test scenario state will be built. Following the test scenarios, the testing simulation will be performed, in which the interaction between the software and human will be realized by applying the action based on wizard human's mental rule. Result and experiences of testing simulation process will be recorded by the testing simulation monitor. Based on testing result and experience record, information visualization will provide help to further human factor analysis of system behavior description.

To achieve the work describe above, we will take the following steps:

1) Create test scenario generator from the system behavior specification
2) Build a wizard human agent for system simulation by implanting its mental rule, which is abstracted from the human interaction description in the system requirement specification.
3) Build a human error generator based on the system behavior description and the action performance rule for the interaction style.
4) Build the testing simulation environment, perform the testing simulation following the test scenario generated from (1), interact with Wizard Human Agent build from (2), inject the human interaction error generated from (3), and record all the result and process of simulation.
5) Provide graphic display of testing results and the suggestions for further human factor analysis of testing.
6) Perform a domain analysis for a simple application.

C.3.5 Evaluation factors

C.3.5.1 Reduction in Development and Maintenance Time

A reduction in development time is the main reason to provide the automatic engineering facilities to software development. Base on precise and complete description of system behavior, the formal model checking can be done automatically and can verify the software interaction with interactive constrains, such as concurrency, real time constrains. The automatic verification of interaction behavior will greatly reduce the errors in the software design, which will eventually reduce the development time and maintenance time for re-design of software.

By building the Wizard Human Automatic Testing Simulation, most of interaction behavior can be tested without real human operating. Most interaction errors will be find out with the completely execution of interaction behavior autonomously. It will greatly help the error elicitation before the real system testing. Eventually, much reduction of system development and support can be achieved.

C.3.5.2 Reduction in Test Cost

The interaction activity in interactive system is the one of the main functionality. To valid and verify the interaction behavior, pass efforts are concentrated on the system testing. System performance testing has several drawbacks. First, cause of the complex design of system, the system behavior stat is tremendous. The complete testing of interaction behavior is impossible. Normal solution to this is to select critical part of system to perform the real test. But, this incomplete testing cannot find the behavior fault during the system longer performance. Also the propagation of interactive error will omit. This incomplete testing cannot give guarantee of dependability of system. For Second, test is time consuming and high cost. For each test, the human labor and environment setting will make the cost of testing process increase.

In this proposed work, the test scenario is generated after performing model checking of system design. The verification of system design will elicit the system design error for first routine. Based on the verified system design, the number of generated test scenario would be shrink. This would highly reduce the test cost from the test cases number aspect. In addition, the proposed substitution of human with wizard human agent greatly decreased the cost of human testing labor. It will not only reduce the cost of human test labor, but also improve the efficiency of test simulation. Although the generation of the action rule of wizard human should be done in additional, the benefit of wizard human test simulation is greatly cover the extra effort. So, with two steps facilities, the reduction in test cost is achieved.

C.3.5.3 Improvement of Dependability based on the Interaction Attribute Specification

The requirement of interactive system has not been considered completely in the pass research. The less study of interaction requirement will feed the hidden danger for system design. By providing the specification of interactive requirement, the concurrency, real time properties of system would be measured during the system design and verification. This would increase the insurance of correctness of system design. The definition of the interactive measurable attribute make the system's interaction can be quantitative evaluated

C.3.5.5 System Evolution

Interactive system is becoming more and more complex. The behavior of system is a dynamic and uncertainty. The system mostly is built on the qualitative description of behavior and human labor enumerative testing. This make the evolution of the interactive system is extremely difficult. With proposed framework for interactive system development framework and tools, system behavior will be explicitly specified in the system requirement analysis phase. The measurable attribute of system interaction will be implicated in the system design. The insurance of the interaction behavior in the system design and the implementation with the behavior specification will automatic done. For the interaction system with these facilities, the new interaction requirement and feature is easy to added into the former system specification. The consistence of interactions and the additional side effects will be automatically checked and insured. The tools for software design and implementation would help to perform the system evolution, which would include changing functions, adding modules, improving the attribute requirement and etc.

C.3.6 Schedule

1) Study the current requirement specification methods and their focalizations; investigate appropriateness of methods for using in the human-software interactive system specification. (6 months)
2) Study the description of interaction behavior and related attributes, such as real time and concurrency, and define the constraints, such as safety, reliability, and availability, for interaction system specification. (12 months)

3) Modify the current software prototype system for capability of modeling the interactive character in the system. (3 months)
4) Modify the current software prototype system for capability of illustrating the system attribute for both functional evaluation and nonfunctional evaluation. (3 months)
5) Study the model abstraction method for software behavior extraction with related attributes of interactive system, and select and apply appropriate modeling checking method for the interactive properties. (6 months)
6) Extend the capability of prototype system by including the test scenarios generation from the system specification. The scenarios generation method will be studied based on the interactive characters, such as interactive time constrain, concurrency, and so on. (6 months)
7) Generate human errors for fault injection in the testing based on the human behavior specification and the possibility of action selection designed in the software. (6 months)
8) Creation of the WHAT-FITs environment by building the wizard human agent, which performed based on the mental rule abstracted from the behavior specification of system with human related aspect. (6 months)
9) Graphic result illustration of counter-example of model checking and the result of testing simulation in the prototype interactive system. (3 months)

C.3.7 Comparison with other research

The dependable systems include autonomous system and combination of human operated and automated systems. The interaction are being more and more important for safety-critical system, for example aircraft traffic control system, the failure of which can cause injury or death to human beings. Consequently, the software of these interactive systems needs a high level of dependability. For interactive system, the complexity of the system always seems makes the confusion of what the automat are doing potentially comes out, which greatly harm the quality of interactive system. The reason of proliferation of mode confusion in interactive system is the inconsistency between software system activity mode and human activity mode. The deeper reason of mode inconsistency is that how software really performs (software implementation) cannot fit well with what user want software to do (interaction requirement). So, during the interactive software development procedure, the precisely mapping of interaction requirement to the software implementation is the bottleneck to improve the dependability of interactive system. Furthermore, the most important chains in the mapping of requirement to implementation are requirement elicitation, design verification and implementation validation.

Work related to this topics discussed in this paper includes research in the areas of interaction requirement specification, software design model extraction and verification, human error fault injection, and requirement-based automatic testing.

The interactive system is a kind of hybrid system. It includes not only function requirements, which are the functional correctness of interaction behaviors, but also include the nonfunctional properties, which include the safety, reliability, availability, efficiency and usability. For insurance of both functional requirement and non-functional requirement, it is impossible to consider them separately. For example, it is infeasible to measure the system availability without consideration of behavior correct. So, it is necessary to provide the non-functional attributes descriptions of interactive system at the beginning of the system specification, which normally focus on the functional description. The specification of measurable attributes of system will not only provide precisely criteria for system design, but also help to verify the consistency between the system design and implementation with the system requirement. So, the requirement specification of interactive system should include the descriptions of not only behaviors, but also properties. Passed works of related software specification have decentralized the focus. Davin and Mieke used hybrid automata to description the discrete and continuous properties of hybrid interactive system [44]. Francis Jambon had tried to illustrate the safety and usability by using B formal method [37-38]. Nancy provided intent specification to capture the design rational and assumption made through out the design process, and place great effect on the requirement specification and developed the SpecTRM methodology and its requirements specification method SpecTRM-RL for the safety of interactive system. [3-8].Ozols also performed the safety requirement modeling by using state machine graphs [28]. Paul Curzon concentrated the work on the formalization of correctness and usability proof by using finite state machine description [45]. Luqi had provided the time constrains related software functional specification [52,53]. Harrison had explored interactors to structure statecharts for interaction [19-22]. John rushby described the interactive properties by using MurØ specification [33-

35]. The proposed work will analysis the properties of interactive system, and provide an complete specification of behavior for both software and human, and provide the properties specification for further software design and implementation.

The model checking has been proved to be an efficient method to check the system state transmission with property [14][15]. Model checker like SPIN [46], PROD [47], SMV [48,49], and FDR [50] are commonly used. And the application of model checking in the interactive system is also performed in past few years. John Rushby used model checking MurØ to discover consistence of two described behavior based on the mental model suggested by human factor experts [33-35]. Bruno Blaskovic used SPIN and PROD to verify the communication process [16]. Ozolos perform the verification by carrying out interactive proofs to meet for the state machine. Atlee and Gannon used SMV to verify the timing requirement of system [27]. Harrison used model checking SMV to verify the interface model changes of interactive system [19-22]. Gerald and Victor also did work to analyzing mode confusion by comparison of applying several model checking methods [17]. In this proposal work, we will not only study on the model checking for the time property, safety properties, but also will check the properties for specified interaction techniques, such as undo mechanism, which will greatly affect the system safety and dependability.

Further validation of interactive system implementation is to ensure the prototyped interactive system to be performed under the beginning assumption, especially human behavior integrated overall system performance. Dennis discussed the real time property monitor based on the requirement specification [55]. Our WHAT-FITs test environment will validate the interaction properties based on the system behavior specification. To improve the efficiency of testing, our proposed work include building a wizard human to perform the human actions based on the requirement specification, so that to make the testing automatic and reduce the human labor for testing. Also, during the testing, fault injection is one of important methods to evaluating the dependability of computer systems with inject the intentional design fault [30]. Jeffrey Voas used software fault injection to simulate human operator error scenarios to determine the occurrence [33]. For interactive system, the most dangerous trigger is not only the software fault, but also the human error. Our proposed work will generate human errors based on the behavior of wizard human to perform human fault injection. The automatic of human error generation and the human action performance will greatly increase the testing efficiency and the confidence achieved.

C.4 Broader impacts

If the dependable interactive software can be automatically generated, verified, and validated then the following can take place:

1) The misunderstanding between the interaction requirement and the software design will be eliminated as the design model will be verified and the software implementation will be validated with the interaction specification of the system.
2) We will be able to eliminate the fault of interactive system operations during the software design phase and implementation phase.
3) Safety of the system will be improved as the human behavior is considered in the interactive system development process, consequently reduce the possibility of the software disasters will be affect the human life.
4) Reliability of the interactive system will be improved with the formal model checking of software design model and testing of the software implementation.
5) The costly process of exhaustively human labor testing each of the interaction scenarios can be phased out, as the proposed framework will perform the interaction testing automatically with simulated wizard human action.
6) The time for further human behavior analysis of software implementation will be reduced, as the wizard human has acted as the human to check the consistence of software design behavior.
7) The future analysis of the interaction system requirement and training planning will be possibly performed with the recorded data of testing experience and software operation sequences.

C.4.1 Transition of technology

Technology transfer will be addressed by integrating the proposed interactive software development framework and tools with existing prototype system development. By re-using and extending successfully applied PSDL language, extending the current prototype system design with interaction model checking, and creatively building the wizard

human automatic fault inject testing simulation, the general acceptance of our approach is enhanced. Also, publishing result in OMG, ACM, and IEEE sponsored conference and make extended prototype system available can facilitate acceptance.

The Software Engineering Automatic Center at the Naval Postgraduate School offers M.S. and Ph.D. degrees. The students at NPS will contribute to this research and development effort. Their involvement will facilitate information transfer into the DoD further. We also plan to integrate emerging technologies into the courses we teach.

C.4.2 Experimentation and integration plan

The work will be performed by the faculty of the Software Engineering Automatic Center at the Naval Postgraduate School and their Ph.D. and M.S. students. The principle investigators will be responsible for coordination of the following plan previously stated in section 3.6 for schedule:

1) Study the current interaction requirement specification methods, which are mostly special purpose designed. Investigate appropriateness of methods for using in the human-software interactive system specification.
2) Create the extension of current design modeling of modeling, with explicitly definition of the measurable metrics for interactive system.
3) Modify the current software prototype system for capability of modeling the human character in the system.
4) Modify the current software prototype system for capability of illustrating the system attribute for both functional evaluation and nonfunctional evaluation.
5) Select proper attributes of interactive system, and select appropriate modeling checking method for the interactive properties.
6) Extend the capability of prototype system by including the test scenarios generation from the software model. The scenarios generation method will be studied based on the interactive characters, such as interactive time constrain, concurrency, and so on.
7) Creation of the WHAT-FITs environment by building the wizard human agent which performed based on the mental rule abstracted form the model specification of human related aspect.
8) Generate human errors for fault injection in the testing based on the human behavior features and the possibility of action selection designed in the software.
9) Analysis and interpretation of the result and experience of simulation. Further analysis and interpretation principle will be studied, according to which further suggestions can be given out.
10) Graphic result illustration of counter-example of model checking and the result of testing simulation in the prototype interactive system.

REFERENCES CITED

1. *Paloma Diaz, Ignacio Aedo, & Fivos Panetsos,* ***Modeling the Dynamic Behavior of Hypermedia Applications,*** *IEEE transaction on software engineering, Vol. 27, No. 6, June 2001*

2. *Gerard J. H. and Margaret H.,* ***An automated verification method for distributed system software based on model extraction,*** *IEEE transaction on software engineering, vol.28, No.4, April 2002*

3. *Mario Rodriguez, Marc Zimmerman, Masafumi Katahira, Maxime de Villepin, Benjamin Ingram, and Nancy Leveson, Identifying Mode Confusion Potential in Software Design, Digital Aviation Systems Conference, October 2000.*
4. *Nancy Leveson, Intent Specifications: An Approach to Building Human-Centered Specifications, IEEE Trans. on Software Engineering, January 2000*

5. *Nancy G. Leveson, L. Denise Pinnel, Sean David Sandys, Shuichi Koga, Jon Damon Reese. Analyzing Software Specifications for Mode Confusion Potential, Presented at the Workshop on Human Error and System Development, Glascow, March 1997.*

6. *Nancy G. Leveson and Everett Palmer (NASA Ames Research Center). Designing Automation to Reduce Operator Errors. In the Proceedings of Systems, Man, and Cybernetics Conference, Oct. 1997*

7. *Molly Brown and Nancy G. Leveson, Modeling Controller Tasks for Safety Analysis, Presented at the Workshop on Human Error and System Development, Seattle, April 1998*

8. *Mario Rodriguez, Marc Zimmerman, Masafumi Katahira, Maxime de Villepin, Benjamin Ingram, and Nancy Leveson* ***.Identifying Mode Confusion Potential in Software Design Digital Aviation Systems Conference,*** *October 2000*

9. *Laurence Rognin, Jean-Paul Blanquart,* ***Impact of Communication on Systems Dependability Human Factors Perspectives,*** *SAFECOM'99, LNCS 1698, pp. 113-124, 1999*

10. *Lucia Bilela Leite Filgueiras,* ***Human Performance Reliability in the Design-for-Usability Life Cycle for Safety Human-Computer Interfaces,*** *SAFECOM'99, LNCS 1698, pp. 79-88, 1999*

11. *Jin Mo and Yves Crouzet, A Method for Operator Error Detection Based on Plan Recognition, SAFECOM'99, LNCS 1698, pp. 125-138, 1999*

12. *Rogerio de Lemos and Amer Saeed,* ***Safety Analysis Techniques for Validating Formal Models During Verificaiton,*** *SAFECOM'99, LNCS 1698, pp. 58-66, 1999*

13. *Atif M. Martha P. and Mary Lou S.,* ***Hierarchical GUI Test Case Generation Using Automated Planning,*** *IEEE transactions on software Engineering, vol.27, No.2, Feb. 2001*

14. *B.R. Haverkort; H. Hermanns; J.-P. Katoen:* ***On the use of model checking techniques for dependability evaluation,*** *19th IEEE Symposium on Reliable Distributed Systems, pp.228--237, Erlangen, Germany, October 16--18, 2000*

15. *F. Schneider, S. M. Easterbrook, J. R. Callahan and G. J. Holzmann,* ***Validating Requirements for Fault Tolerant Systems using Model Checking,*** *Third IEEE Conference on Requirements Engineering, Colorado Springs, CO, April 6 - 10, 1998.*

16. *Bruno Blaskovic, Petar Knezevic, and Mirko Randic,* ***Model Checking Approach for Communication Procedures Validaiton,*** *IEEE , pp 532-535, 2001*

17. *Gerald Lüttgen , Victor Carreño,* ***Analyzing mode confusion via model checking,*** *NASA/CR-1999-209332 ICASE Report No. 99-18, Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23681-2199, pp. 67, May 1999.*

18. *Steven P. Miller and James N. Potts,* ***Detecting Mode Confusion Through Formal Modeling and Analysis,*** *NASA/CR-1999-208971, pp. 69 January 1999*

19. *J. Creissac Campos & M. D. Harrison,* ***Model Checking Interactor Specifications.*** *Automated Software Engineering, 8(3/4):275-310, August. 2001*

REFERENCES CITED

20. *Pocock, S., Harrison, M., Wright, P & Johnson, P.* ***THEA: a technique for human error assessment early in design.*** *To appear Proceedings Interact'01, Japan. 2001*

21. *Loer, K. & Harrison, M.D.* ***Formal interactive systems analysis and usability inspection methods: two incompatible worlds?*** *In Palanque & Paterno (Eds) Interactive Systems: Design, Specification and Verification. Springer Lecture Notes in Computer Science 1946. pp. 169-190. 2001*

22. *Willans, J, and Harrison, M.D. (in press) A toolset supported approach for designing and testing virtual environment techniques. International Journal of Human Computer Studies.*

23. *Smith, S.P., Duke, D.J. & Massink, M.* ***The hybrid world of virtual environments.*** *Computer Graphics Forum. 18(3) C297-307. 1999*

24. *James Corbett, Matthew Dwyer, John Hatcliff, Robby, Expressing Checkable Properties of Dynamic Systems: The Bandera Specification Language, June, KSU CIS Technical Report 2001-04. 2001.*

25. *Randell, B. 2000.* ***Facing up to faults.*** *The Computer Journal, vol 43, pp. 95-106, 2000.*

26. *Ricky W. Butler, etc,* ***a formal methods approach to the analysis of mode confusion,*** *http://shemesh.larc.nasa.gov/fm/papers/butler-etal-dasc98.pdf*

27. *J. M. Atlee and J. Gannon,* ***Model Checking Timing Requirements,*** *Computer Science Technical Report CS93-25, University of Waterloo, September 1996.*

28. *M. A. Ozols, K. A. Eastaughffe, A. Cant, S. Collignon.* ***DOVE: A Tool for Design Modelling and Verification in Safety Critical Systems,*** *Proceedings of the 16th International System Safety Conference, Seattle, US September 1998.*

29. *B. Blaskovic, P. Knezevic, M. Randic, University of Zagreb, Zagreb,* ***Croatia, Model Checking Approach for Communication Procedures Validation,*** *EUROCON'2001, pp.532-535*

30. *Mei-Chen Hsueh, Timothy K. Tsai, Ravishankar K. Iyer,* ***Fault Injection Techniques and Tools,*** *IEEE Computer, April, (Vol. 30, No. 4), pp. 75-82, 1997*

31. *Gerald Lüttgen, Victor Carreño,* ***Analyzing Mode Confusion via Model Checking.*** *120-135, LNCS 1680, pp. 120*

32. *Jeffrey Voas,* ***Analyzing Software Sensitivity to Human Error,*** *Failure & Lessons Learned in Information Technology Management, Vol. 2, pp. 201-206, 1998*

33. *John Rushby,* ***Modeling the Human in Human Factors,*** *Volume 2187. SafeComp 2001: Proceedings of the 20th International Conference on Computer Safety, Reliability, and Security}.Edited by Udo Voges., Springer-Verlag, Budapest, Hungary., September, pp. 86–91, 2001.*

34. *Judy Crow, Denis Javaux (University of Liege), and John Rushby,* ***Models and Mechanized Methods that Integrate Human Factors into Automation Design,*** *HCI-AERO 2000: International Conference on Human-Computer Interaction in Aeronautics. Edited by Kathy Abbott and Jean-Jacques Speyer and Guy Boy. Toulouse, France. September, 2000. Pages 163–168.*

35. *John Rushby,* ***Using model checking to help discover mode confusions and other automation surprises,*** *The 3rd Workshop on Human Error, safety, and system development (HESSD'99), Liege, Belgium, 7-8 June, 1999.*

36. *Paul E. Ammann, Paul E. Black, and William Majurski,* ***Using Model Checking to Generate Tests from Specifications,*** *Proceedings of 2nd IEEE International Conference on Formal Engineering Methods (ICFEM'98), Brisbane, Australia (December 1998), edited by John Staples, Michael G. Hinchey, and Shaoying Liu, IEEE Computer Society, pages 46-54.*

37. *Francis Jambon.,* ***From Formal Specifications to Secure Implementations.*** *Computer-Aided Design of User Interfaces (CADUI'2002), edited by Kolski, Christophe and Vanderdonckt, Jean, Valenciennes, France, Kluwer Academics, 2002, pp. 43-54*

REFERENCES CITED

38. *Francis Jambon, Patrick Girard and Yamine Aït-Ameur.,* ***Interactive System Safety and Usability enforced with the development process.*** *Engineering for Human-Computer Interaction (8th IFIP International Conference, EHCI'01, Toronto, Canada, May 2001), vol. 2254, Lecture Notes in Computer Science, edited by Little, Reed Murray and Nigay, Laurence, Berlin, Springer, 2001, pp. 39-55*

39. *Ian MacColl, David Carrington,* ***Specification-Based Testing of Interactive Systems (1997),*** *March 20, 1997, http://archive.csee.uq.edu.au/~ianm/*

40. *D.Clarke, I.Lee.* ***Testing Real-Time Constraints in a Process Algebraic Setting.*** *In Proceedings of the 17th International Conference on Software Engineering, 1995.*

41. *Gordon D. Baxter, A. Monk et al,* ***It's about time: Reasoning about timing requirements in human-machine systems,*** *Berlin 2001, Springer, http://arjuna.ncl.ac.uk/publications/*

42. *D. Leadbetter, A. Hussey, P. Lindsay, A. Neal and M. Humphreys.* ***Towards Model Based Prediction of Human Error Rates in Interactive Systems.*** *Australian Computer Science Communications: Australasian User Interface Conference 2001, 23(5), pp.42-49, 2001.*

43. *D. Navarre, P. Palanque, R. Bastide and O. Sy.* ***A Model-Based Tool for Interactive Prototyping of Highly Interactive Applications.*** *12th IEEE, International Workshop on Rapid System Prototyping , Monterey (USA). IEEE , 2001.*

44. *G.Doherty, M. Massink, G. Faconti* ***Using Hybrid Automata to Support Human Factors Analysis of a Critical System,*** *Formal Methods in System Design 19 (2) S. Gnesi and D. Latella (Eds), Kluwer Academic Publishers, (September 2001)*

45. *P. Curzon and A. Blandford,* ***Using a Verification System to Reason about Post-Completion Errors.*** *Participants Proceedings of DSV-IS 2000: 7th International Workshop on Design, Specification and Verification of Interactive Systems, at the 22nd International Conference on Software Engineering, Edited by Philippe Palanque and Fabio Patern? pages 292-308, June 5-6 2000, Limerick, Ireland.*

46. ***ON-THE-FLY, LTL MODEL CHECKING with SPIN,*** *http://netlib.bell-labs.com/netlib/spin/whatispin.html*

47. ***PROD 3.3.09, An Advanced Tool for Efficient Reachability Analysis,*** *http://www.tcs.hut.fi/Software/prod/*

48. *Ken McMillan,* ***Lecture notes for NATO summer school on verification of digital and hybrid systems,*** *http://www-cad.eecs.berkeley.edu/~kenmcmil/tutorial/toc.html*

49. ***The SMV home page.*** *smv is installed locally on the Suns (e.g., dipsy), in /bham/ums/solaris/pd/packages/smv/smv. The examples are in /bham/ums/solaris/pd/packages/smv/examples/.*