TESTING A SAFETY-CRITICAL APPLICATION†

John C. Knight, Aaron G. Cass, Antonio M. Fernández, Kevin G. Wika

Department of Computer Science
University of Virginia
Charlottesville, VA 22903

Contact Author:

John C. Knight
Department of Computer Science
University of Virginia
Charlottesville, VA 22903

(804)982-2216
knight@Virginia.EDU

†. Supported in part by the National Science Foundation under grant number CCR-9213427 and in part by NASA under grant number NAG1-1123-FDP.

Abstract

For a safety-critical system, a system whose consequences of failure might be very high, it is not possible to rely upon testing to provide the necessary verification. Confidence in the software has to be achieved by a variety of verification techniques. Nevertheless, testing can yield valuable information about a system, and those who will eventually use it expect a system to have been tested.

As part of a research program, we are developing the software for an experimental safety-critical application and we have begun consideration of how the software should be tested. Of particular interest are the issues of the role of testing in the verification process and techniques for testing complex safety-critical systems.

The target safety-critical application provides many challenges for a test system. The application software is quite complex, being based on a distributed architecture and utilizing significant amounts of "off-the-shelf" software. Both of these architectural decisions were essential to meet the functional requirements of the system. Complexity also comes from the application devices that include a complex X-ray imaging system and high-energy electrical devices that cannot be made available for testing on a routine or extended basis. Any testing strategy must also account for the interactive nature of the system and the need for extensive operator direction. Finally, correct operation of the system is difficult to assess because a variety of complex calculations are required to perform the necessary control.

In this paper, we describe the approaches we have developed to system-level testing of this application. We discuss the overall test harness structure including the mechanisms for synthetic input generation from both the operator and the peripheral devices. Error detection is based on systematic application of reversal checks throughout the software. The potential benefits and difficulties with this approach are discussed. Finally, we present our approach to test case selection. The goal of the selected test cases is to permit rigorous, if narrow, results to be concluded about the software with feasible numbers of test cases. In this way we hope to establish useful system-level properties by testing.

I Introduction

Computing systems in which the consequences of failure can be very high are termed *safety-critical*. Many such systems exist in application areas such as aerospace, defense, transportation, power-generation, and medicine. Public exposure to these safety-critical systems is increasing rapidly, and, since the correct operation of these systems depends on software, the possibility of serious damage resulting from a software defect is considerable and growing.

In most cases, it is not possible to rely upon testing to provide the necessary verification of the software in a safety-critical application. Testing aimed at providing assurance that a statistical dependability metric (such as mean time to failure) has been achieved is infeasible [3]. The possibility of a failure going unobserved during testing also exists and this increases the difficulty of using testing as a verification technique [1]. Nevertheless, testing can yield valuable information about a system, and those who will eventually use a system expect it to have been tested.

We are engaged in a case study in software development in which we are developing the software for an experimental safety-critical application. The application is in the medical domain with the potential for human injury or equipment damage in the event of failure. Although we intend to exploit a variety of approaches to verification as development proceeds, we have begun consideration of how the software should be tested. Our research goals in the area of testing are: (a) to determine what role testing should play in the development of such a system; (b) to determine what rigorous conclusions can be drawn about the software through a test process; and (c) to determine how to test a relatively complex system in the most effective way.

The issues raised by the testing of this application are considerable for the following reasons:

- Despite the extremely undesirable complexity that it introduces, the system uses a distributed architecture and depends upon significant amounts of “off-the-shelf” software that is not under our control.
- The application uses a complex X-ray imaging system and high-energy electrical devices that cannot be made available for testing on a routine or extended basis.
- The system is interactive and requires extensive operator direction via a graphic user interface. The “operator” in this case is a physician.

- Correct operation of the system is very difficult to determine because a variety of complex calculations are required to perform the necessary control. The application has all the appearances of being untestable [4].

In this paper, we describe the approaches we have developed to system-level testing of this application. In the next section, we summarize the application and describe the key issues faced in testing the software. In section III, we discuss the overall test harness structure including the mechanisms for synthetic input generation from both the operator and the peripheral devices. The error detection mechanism is described in section IV. In section V, we discuss our approach to test case selection and in section VI we present our conclusions.

II The Application

The case study in which we are presently engaged is *The Magnetic Stereotaxis System* (MSS). This is an investigational device for performing human neurosurgery being developed in a joint effort between the Department of Physics at the University of Virginia and the Department of Neurosurgery at the University of Iowa [5].

The system operates by manipulating a small permanent magnet (known as a “seed”) within the brain using an externally applied magnetic field. By varying the magnitude and gradient of the external magnetic field, the seed can be moved along a non-linear path and positioned at a site requiring therapy, e.g., a tumor. The magnetic field required for movement through brain tissue is extremely high, and, in the MSS, the required field is generated by a set of six superconducting magnets that are located in a housing that surrounds the patient’s head. Fig. 1 shows how the system is organized.

A key element of the device is the imaging subsystem. It uses two X-ray cameras positioned at right angles to detect in real time the locations of the seed and of X-ray opaque markers affixed to the patient’s skull. The X-ray images are not displayed. Instead, they are processed by the imaging subsystem so as to locate the objects of interest in a canonical frame of reference, and this information is used to display graphic representations of the seed and skull markers on pre-operative magnetic resonance (MR) images. The MR images are the primary source of information used by the surgeon for making control decisions.

The MSS has the potential for being used for hyperthermia by radio-frequency heating of the seed from an external source or for chemotherapy by using the seed to deliver drugs

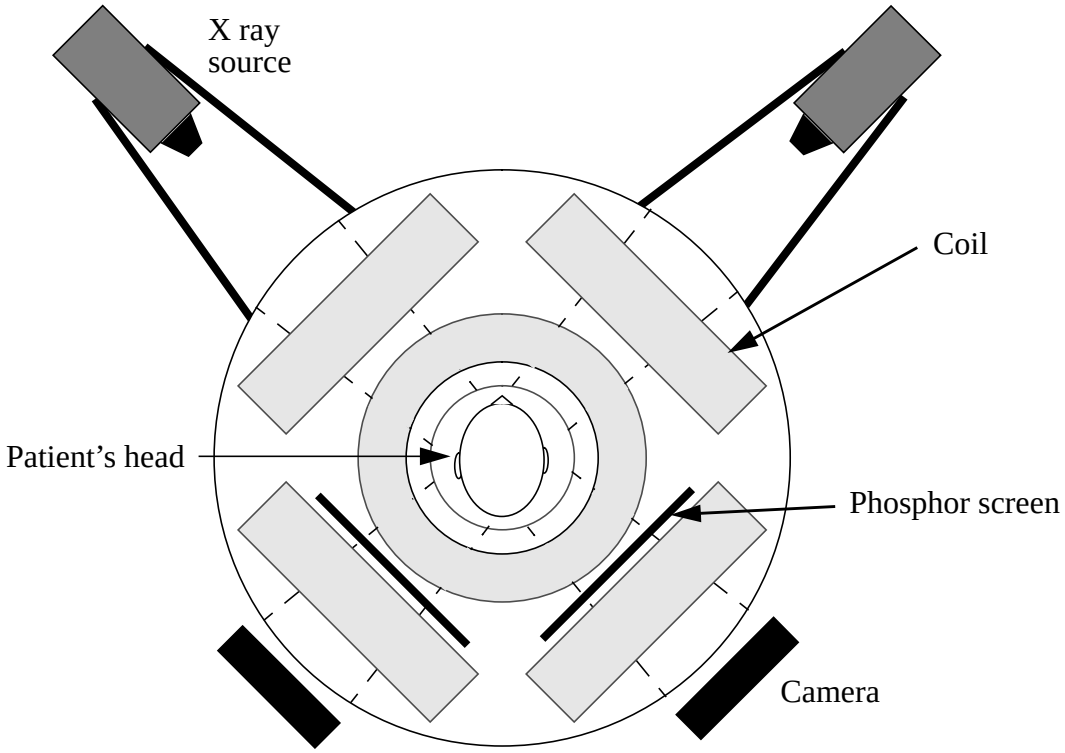

Fig. 1. View of MSS from above patient's head.

to a site within the brain. The MSS concept promises to be far less traumatic to the patient than present invasive approaches to such treatments. The state of the MSS is that the concept is fully defined, the majority of the basic research in physics is complete, and a fully functional prototype is being used for demonstration and evaluation.

Failure of the MSS control software could have serious results. The most obvious problem that comes to mind is a software defect that moves the seed in an undesired manner and damages critical brain tissue. Less obvious is the possibility of one of the six electromagnets or an associated power supply failing so that the magnetic field suddenly becomes distorted. In this case, the software is expected to act quickly to shut down the remaining magnets in order to stop seed motion and prevent patient injury. Other relatively obscure but very serious problems include the fact that the entire system depends on the correct operation of the imaging system which itself depends on the X-ray cameras being correctly calibrated. Calibration is a complex software function and a small error in calibration leads to a large error in position estimation. Any such error leads to incorrect information being displayed to the surgeon, and the possibility of subsequent harm to the patient is high. Finally, there are regions of the magnetic field in which the seed can accelerate once motion begins making appropriate control very difficult. This has to be detected

and prevented by the software.

With these and many other possibilities for harm, and the entire system depending on correct software, we began the case study with the assumption that an extremely simple software system with minimal functionality would be used. The simpler the system the better the chance of getting it right. In practice, however, a simple system is out of the question because the application requires several high-resolution graphic displays and these depend upon extensive computation for data generation. Present displays include the operator display containing the MR images and various control panels, a field display that provides a visualization of the magnetic field updated in real time to show the field being applied to the patient, and an engineering display that presents status information for use in monitoring the system. In addition, although not presently implemented, a three-dimensional display of the brain reconstructed from a set of MR "slices" is expected to be needed before the system can be used routinely for human therapy.

This demand for computation and display has forced the use of a distributed architecture. This in turn has forced the use of a substantial amount of "off-the-shelf" software to perform routine operating system and network functions. How such a system will be verified or even if it can be is a complex issue that we address elsewhere [6]. In this paper, we are concerned with questions of how such a system might be tested.

The overall software architecture of the system is shown in Fig. 2. It consists of a control program that interfaces with the various peripherals and the display programs. Communication between these programs is over a local-area network. Each program executes on a separate computer running Unix and the network links are Unix socket connections. The graphic user interfaces are implemented using X Windows.

The key issues facing us in testing this system are:

- How can test case execution be automated and operated in a self-contained, closed-loop fashion? The system is interactive and expects the surgeon to enter requests by depressing buttons, selecting objects, and manipulating input widgets (e.g., "sliders") on a graphic user interface. In addition, the images come from a complex imaging system built of special purpose hardware. The cameras yield X-ray images that are noisy, suffer from severe radial distortion, and have significantly different exposures at the center of the image than the edges.

- What test inputs should be used? There are, of course, an arbitrary number of pos-

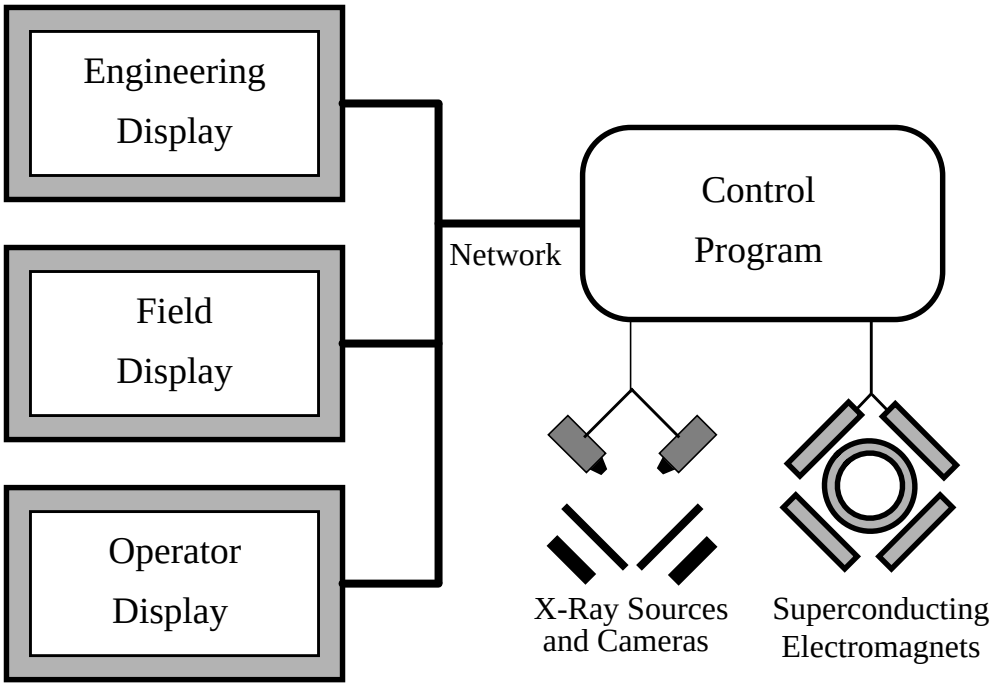

Fig. 2. Distributed MSS software architecture

sible combinations of seed positions, desired moves, etc.

- How can we determine whether the “outputs” are correct?
- Since the concern here is safe operation rather than availability or reliability, what role, if any, can testing play in rigorous verification that safety requirements are being met?

In general, this system poses significant challenges in the area of system testing. We anticipate that the experience gained in addressing them will yield valuable insights into testing complex systems.

III The Test System Structure

Important goals in any system-level testing activity are to have the entire software system operating and to disturb the software as little as possible. In testing this system, we have to be able to execute tests with all of the subsystem programs running, with the network operating, and with image and operator input data from the expected operational profile. We also have to be able to execute the software with as little change as possible from its operational form.

There are many different forms of test case that will be executed. One of the most important and elaborate is a complete surgical procedure. Such a test case consists of calibration of the imaging system followed by an arbitrary number of moves of the seed within the operating region. To support this and other forms of system test, we have developed a test system structure that is shown in Fig. 3 and consists of the following:

- the complete MSS software system with two modifications, and
- a test harness that is responsible for selecting test cases and sequencing test events, and that contains an image generation subsystem that produces synthetic X-ray images and a simulator for the superconducting coils.

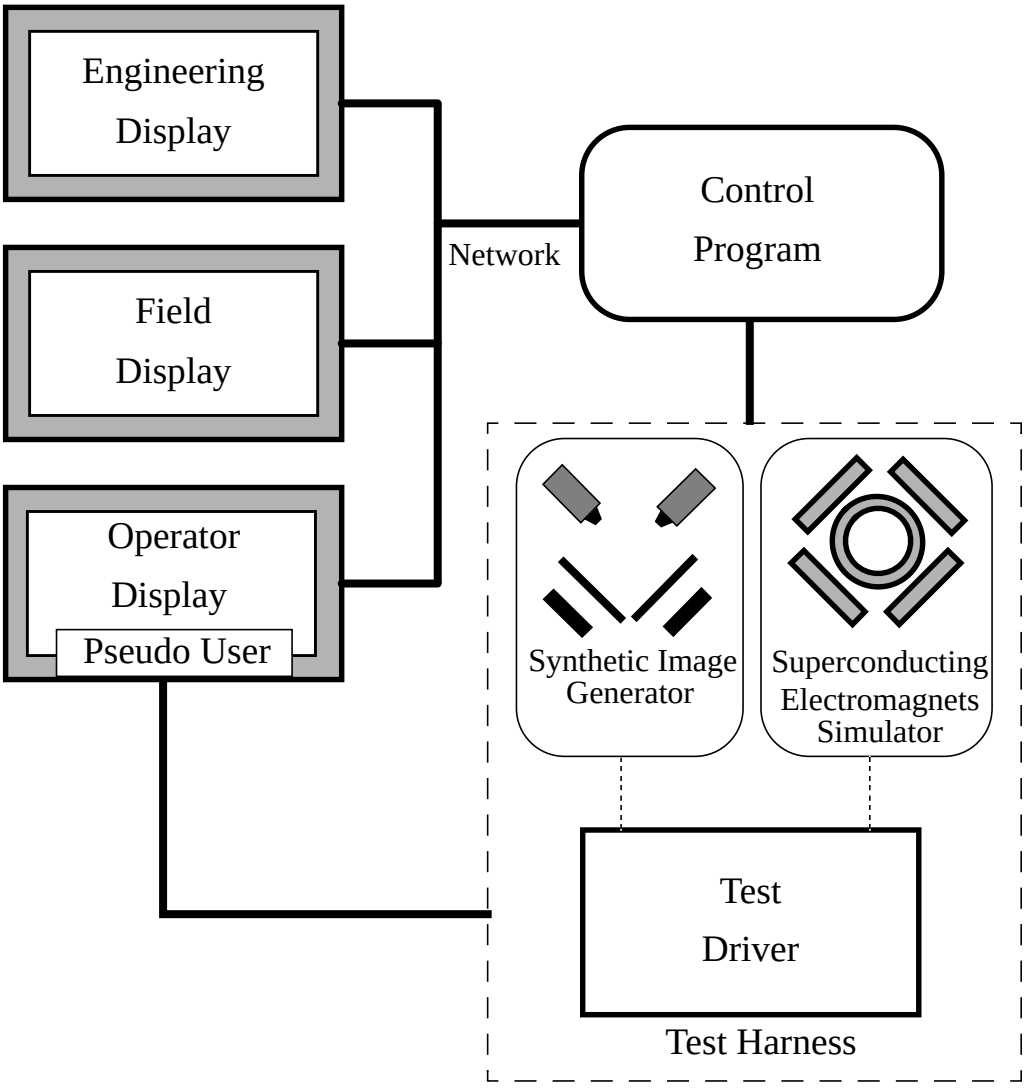

Fig. 3. System architecture with test harness

The first modification to the software system was the replacement of the device drivers for the X-ray system and superconducting coils so that test inputs and outputs could be dealt with. The second modification was the addition of a small module in the operator display to permits synthetic operator commands to be entered.

The test harness operates as a separate program on a different computer from the rest of the system. It communicates with the software under test via two socket connections. One of these connections is used to transfer synthetic images to the replacement X-ray device driver. The second is used to transmit operator commands that the test harness determines to be part of a test case to the operator display. The most significant aspects of the test harness are the image generation and operator command entry systems, and they are described below.

Image Generation System

Our initial approach to image generation was to build hardware to provide realistic digital images. This hardware operated with visible light rather than X rays and was built to 1/2 scale. High-precision placement stages were used in all three dimensions to locate the objects in positions known to within 1/10 of a millimeter. Testing of the imaging system in the MSS software was then performed by moving the objects using the placement stages.

This approach was highly unsatisfactory for a number of reasons. First, it was painfully slow and completely dependent on human operation. This is not an approach that is conducive to high-volume testing. A second significant reason for rejecting the approach was that the visible light sources and cameras were not able to produce images with the same distortions, backgrounds, and noise levels as the X-ray systems.

The present system uses synthesized images that are produced as follows. The desired object positions in the canonical coordinate system are generated initially by the test harness as part of a test case. The projections of the objects onto the two camera sensing surfaces are then computed and used to place correctly located shadows onto the two digital images. These projections take into account a multitude of deviations from perfect positioning of the real equipment. The X-ray sources, for example, are not located precisely on a line perpendicular to the center of the camera. Once the projections have been generated, realistic distortions are applied to the images, and finally the images are mixed with real backgrounds taken from the real X-ray system to produce extremely high-fidelity synthetic images.

The Control Program's interface with the synthetic image generator is identical to the real X-ray system down to the level of the device driver. The synthetic images arrive over a socket connection whereas the real X-ray images arrive via a custom communications system.

Operator Command Generation

Normally, operator commands are entered using a mouse. Therefore, testing the system under operational conditions would seem to require a human operator. This is unsatisfactory because it is difficult to repeat and because it is very slow.

An approach used in some testing systems is known as "record-playback". Human mouse activity is essentially recorded in a file and the mouse input for a system can be directed to this file to permit mouse activity to be played back. Although useful, this approach does not permit high volume testing because test cases cannot be easily synthesized. Generating suitable mouse actions requires the generator to know the entire geometry of the screen at the pixel level.

Our original approach to dealing with this aspect of testing (which fortunately we did not implement) was to replace the X Windows system with a set of identical facilities that did nothing but keep the rest of the application happy. Thus, requests by the application to draw anything would be acknowledged but ignored. Inputs would be generated at the command level rather than the screen-pixel level, so that realistic inputs could be synthesized faithfully, easily, and in high volume.

Apart from the considerable effort involved in building such as system, it is not a good solution because it fails to test any of the X-Windows software that the application is using. Given the goal of testing as much of the application as possible, this is hardly a satisfactory approach.

Our current approach provides a level of input control that allows command-level inputs from the Test Harness, keeps all the displays operating, and leaves X Windows in place. It operates as follows. A relatively small addition has been made to the Operator Display program that will be used during testing and will remain present but inactive during operation. This addition accepts high-level directions from the Test Harness that are in the form of high-level action requests such as "push the calibrate button". The modification to the Operator Display is referred to as the Pseudo User because that is its role. It transforms the high-level directions either into X events that it injects into the event

queue, or, if necessary, it calls the associated call-back functions.

IV Error Detection

For any reasonable operator request, determining whether the system's action are "correct" is very difficult. For example, the "input" will take the form of a present seed position known up to the resolution of the imaging system, and a direction and distance that the surgeon wishes to move the seed. The "output" will be a set of currents to be applied to the superconducting coils. How can we possibly know whether these are the correct currents? This problem is made significantly worse (if that is possible) by the fact that there are an infinite number of combinations of currents in the six coils that produce the *same* force on the seed.

For this application, we are fortunate in having found what appears to be a workable approach to this problem. The approach is to use *reversal checks* across the entire application. A reversal check is a calculation that takes the output of some computation and regenerates the associated input. If the regenerated input matches the actual input up to the limits imposed by numerical error then there are only two possibilities: either the output is correct or the forward and reverse calculations contain faults that are the *inverses* of each other. Even if the latter were the case, it is unlikely that they would be the *exact* inverses of each other. By exact inverse, we mean that the faults do not permit a failure to be detected on *any* test case whatsoever. In other words, for a fault to go undetected, the forward computation would have to fail in such a way that when its output is used as the input for the reverse computation, that computation fails in such a way that its output is indistinguishable for the original input. And this must occur on every test case for which the system's outputs are in fact wrong. This seems unlikely and so provided sufficient test cases are executed, the probability of detecting the faults can be raised to an acceptable level [2].

We have determined that the entire MSS computation sequence that effects coil control can be covered by reversal checks and we are in the process of implementing them. It is not possible to discuss the details of this coverage here because of space limitations. Instead, we describe the reversal checks used in two major subsystems.

Imaging System

The imaging system accepts as input two X-ray images and locates within them several objects of interest. From the location of the objects in the image and using previously-obtained calibration data, the imaging system determines the location of the objects in the

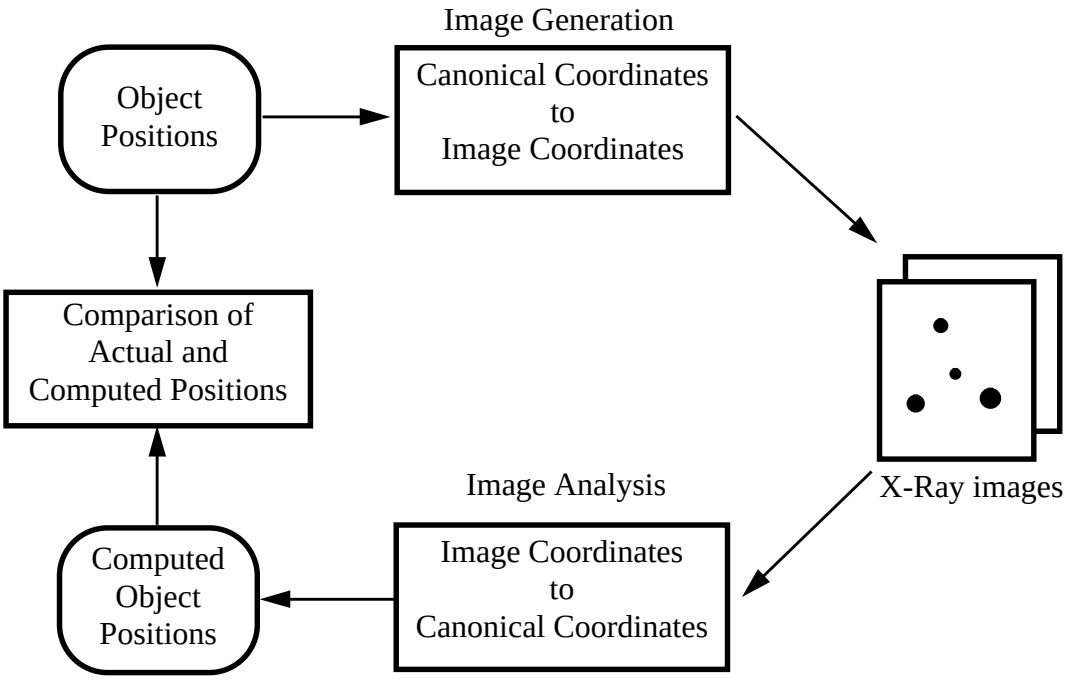

Fig. 4. Reversal check testing of imaging system.

canonical three-dimensional coordinate system.

Recall, however, that in the test system, the images are synthesized. Synthesis is done from object-location data that is generated as part of the test case, and this object-location data is with reference to the canonical coordinate system. Thus, the imaging system is analyzing images in an effort to determine the very data that was originally part of the test case. The correctness check is, therefore, merely to compare the output of the imaging system with the initial test case data. The sequence of steps is shown in detail in Fig. 4.

Coil Current Calculation

The computation of the required currents through the superconducting coils is a surprisingly difficult task. The difficulty arises because the input to the computation is a required force. The output is a six-element vector, the six coil currents, but the currents selected have to be as close as possible to the last currents used in order to minimize the necessary changes. There is no known way to compute the currents in any optimal sense and so various complex approximations are used.

Fortunately, the force produced by a set of six currents passing through coils in a known geometric configuration, i.e. the reverse computation, is easily and exactly com-

putable. Thus, in this case, the reversal check is merely to compute the force that would be produced by the set of six currents determined by the system and compare the value with the desired force.

We find the application of reversal checks to the entire system to be an appealing approach to error detection. Determining rigorously whether the systematic use of reversal checks as an error detection mechanism is indeed an approach with quantifiable benefits is the subject of ongoing research. We note the additional benefit that many of the reversal checks can be (and will be) employed as execution-time assertions. In particular, the reversal check on the current calculations will provide run-time assurance that the difference between the actual and requested force on the seed will be within a specified tolerance.

V Test Case Selection

The test system as described so far is operational and will enable the execution of very large numbers of test cases without human intervention and with considerable confidence in the system's ability to detect failure when it occurs. Given that situation, which test cases should be run?

The research showing that verification of safety-critical systems by testing is infeasible would seem to imply that testing has no real value for safety-critical systems such as the MSS [3]. That result, however, applies to functional correctness. In other words, running sufficient tests to ascertain that a program's output is correct with a high degree of statistical confidence requires an infeasible number of tests. For safety-critical applications, however, we believe that testing to demonstrate useful system properties, no matter how limited, is also valuable and *is* feasible. It is feasible because for many useful properties, what amounts to exhaustive testing can be employed. With the MSS, although we cannot possibly demonstrate functional correctness with a feasible number of tests, we can show some useful if very narrow properties. We give some examples of this idea in this section.

Imaging System

The imaging system is complex and knowing that it locates objects correctly and within required accuracy bounds is a useful property. We believe that this property can be established by high-volume testing. The approach we intend to undertake with the imaging system is to place an object in the field of view and move it systematically throughout

the entire operating region, i.e., raster scan the object through the operating region. Although we speak of objects and motion here, recall that the images are synthesized, and so the raster scanning referred to is in fact performed by a set of nested loops.

Since the images are digital, there is a minimum distance that an object has to move in order for the move to be detectable. If we move the target object at the resolution limit, we are guaranteed to achieve exhaustive testing of the imaging system's ability to determine the position of an object in the field of view. The estimated time required to perform these tests is only on the order of hours.

Although this is a useful property, it is by no means a complete test of the imaging system, and it is important to keep this in mind. This set of test cases does not test, for example, the imaging system's ability to detect objects in the presence of noise, its ability to distinguish between multiple objects, and so on. However, exhaustive testing of the form described above provides assurance that one aspect of image processing is being performed correctly.

Coil Current Calculation

In a similar fashion to the imaging system testing, we can apply exhaustive testing to the coil current calculation to show a narrow but useful property. The current calculation is based on a desired force and present seed location. As noted above, the seed location can be varied systematically over the operating region. During testing, the seed is positioned at small intervals (e.g., 0.5 mm) on a three dimensional grid. During operation of the system, the seed position is always rounded so that it matches these tested locations. In addition, for each possible seed location, a finite number of movement directions and a similarly limited number of force magnitudes have been established. Although generally real-valued quantities, the operational force direction and magnitude are deliberately constrained to be discrete for the express purpose of permitting exhaustive testing. Thus, the coil currents for all possible combinations of seed location, force direction and force magnitude can be computed and the total number of tests is bounded. The estimated time required to perform these tests is on the order of hundreds of hours. The result is that all possible movements that can be requested during a surgical procedure can be evaluated during the exhaustive testing.

The rounding of the seed position and establishment of discrete movement requests are certainly not functional requirements of the system, but are design decisions made to enable exhaustive testing to assure a valuable safety property. The adjustments in the seed

position are on the order of the resolution of the imaging system and are significantly smaller than the spacing of points used to compute the force on the seed. The discretization of the force direction and magnitude is at or below the level of discretization already present in graphics-based input widgets. As a result, these minor restrictions will have no adverse impact on the functional or safety properties of the system while permitting the assurance of a significant safety property.

VI Conclusions

Assurance of dependability in complex safety-critical systems is difficult. No one verification technique is sufficient, and in particular, testing has been convincingly shown to be insufficient for demonstrating functional correctness in nontrivial applications. However, in spite of its limitations, testing can play an important role in demonstrating important properties of safety-critical systems.

We have described an approach to testing a safety-critical system that permits the entire system to be tested without human intervention and with a comprehensive approach to error detection. Closed-loop operation of the test system is accommodated by the use of simulated devices and the introduction of a Pseudo User that provides the means for injecting synthetic operator commands into the system. Error detection is effected through the use of reversal checks. The reversal check should be an effective error detection technique for the MSS because of the improbability of an exact match of the forward and reverse calculation and because of the high volume testing supported by the test system. Although the described system is operational, it has not yet been used to perform any extensive testing. We have established the manner in which the testing will be conducted and the feasibility of the approach.

Test cases identified for the MSS will enable the rigorous demonstration of important properties of the imaging system and current calculation algorithms. The test cases utilize reversal checks for error detection and rely on exhaustive sets of inputs. Exhaustive testing is enabled by appropriate discretization of several real-valued quantities. Although the complete set of safety properties of the MSS (or any other system) will need to be established by a range of verification techniques, we assert that testing will play an important role in demonstrating a subset of safety properties — properties that might otherwise be very difficult to establish with other verification techniques.

Acknowledgments

This work was supported in part by the National Science Foundation under grant number CCR-9213427, and in part by NASA under grant number NAG1-1123-FDP.

References

1. Ammann, P. E., S. S. Brilliant, and J. C. Knight, “The Effect of Imperfect Error Detection on Reliability Assessment via Life Testing,” *IEEE Transactions on Software Engineering*, Vol. 20-2, February 1994.

2. Brilliant, S. S., J. C. Knight, and P. E. Ammann, “On the Performance of Software Testing Using Multiple Versions,” in *Proceedings of Fault-Tolerant Computing: The Twentieth International Symposium*, Newcastle upon Tyne, England, 1990.

3. Butler, R. W. and G. B. Finelli, “The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software,” *IEEE Transactions on Software Engineering*, Vol. 19-1, pp. 3 - 12, January 1993.

4. Weyuker, E. J., “On Testing Non-Testable Programs,” *Computer Journal* Vol. 25-4, November 1982.

5. Wika, K. G., “A User Interface and Control Algorithm for the Video Tumor Fighter,” Masters Thesis, University of Virginia, May 1991.

6. Wika, K. G. and J. C. Knight, “A Safety Kernel Architecture,” Department of Computer Science, University of Virginia, Technical Report No. CS-94-04, February 1994.