REMOTE DIAGNOSIS SERVER[1]

Dr. Somnath Deb, Dr. Sudipto Ghoshal, Venkata N. Malepati and Kevin Cavanaugh,

Qualtech Systems, Inc., 100 Great Meadow Road, Suite 501, Wethersfield, Connecticut 06109

Email: deb@teamqsi.com; Web: http://www.teamqsi.com/rds

Abstract

Modern systems such as fly-by-wire aircraft, nuclear power plants, manufacturing facilities, battlefields, etc. are all examples of highly connected network enabled systems. Many of these systems are also mission critical, and need to be monitored round the clock. Such systems typically consist of embedded sensors in networked subsystems that can transmit data to central (or remote) monitoring stations.

Moreover, many legacy systems were originally not designed for real-time onboard diagnosis, but are safety critical, and would benefit from such a solution. Embedding additional software or hardware in such systems is often considered too intrusive, and introduces flight safety and validation concerns. Such systems can be equipped to transmit the sensor data to a remote-processing center for continuous health monitoring.

At Qualtech Systems, we are developing a Remote Diagnosis Server (RDS) that can support multiple simultaneous diagnostic sessions from a variety of remote systems. The RDS server is built on a three-tier architecture with a "Broker" application in the middle layer, and multiple TEAMS-RT and TEAMATE based reasoners at the backend. The client layer consists of sensor agents that collect test results and transmit them over a message-passing network. The resultant solution is remarkably efficient. Even an old 50MHz Sparc20 can support tens of concurrent sessions involving hundreds of tests. The solution scales easily to hundreds of sessions in any modern workstation or server.

Inspired by the significant interest from the aerospace community, we are enhancing the scalability of our RDS solution to solve huge and complex systems such as system-wide health monitoring of the International Space station or a fleet of commercial jetliners. We also recently won an STTR from NASA to make RDS accessible to automobiles and appliances over standard wireless telephone networks.

Introduction

Continuous on-line *real-time failure detection, and isolation* capability is essential to economical operation of complex systems. Such a capability:

- **Improves operational safety:** Root causes, and their criticality, are quickly and automatically identified, and possibly mitigated
- **Improves availability:** Since most, if not all, the diagnosis is performed online, downtime for troubleshooting is minimized
- **Improves confidence in system serviceability:** The resultant system-wide self-test and monitoring capability enables the health of the system to be continuously and accurately assessed, with a high degree of certainity.

To sum up, a real-time fault detection and isolation solution is essential to faster, cheaper and better operations of complex systems. It also reduces the likelihood of operational failures and disasters due to sudden failures, thereby improving system safety and availability.

In an effort conducted with NASA-ARC in 1996, we had developed a model-based reasoning engine, TEAMS-RT [1,2,3], for the detection and isolation

[1] This research has been sponsored by the Army Research Office (DAAG55-98-C-0057) and NASA-ARC (NAS2-99049).

of multiple faults. The key features of TEAMS-RT are:

- Separation of the system-specific knowledge, captured in terms of models, from the fault-isolation methods. This allows for the same tool to be used on multiple systems using different models.
- Ability to diagnose multiple failures in fault-tolerant systems with multiple modes of operation.
- Ultra-compact memory requirements: only about 2 MB for two subsystems with 1000 failure sources and 1000 test points each.
- Excellent performance using low-end microprocessors (e.g., less than 200ms for the above-mentioned system on a 75MHz Pentium processor).

The TEAMS-RT reasoning engine is therefore an ideal choice for real-time embedded diagnosis. Indeed, we are utilizing it for health monitoring of diverse systems – from Helicopter engine [4] and transmission [5] to 1553 bus systems in the International Space Station [6]. However, embedding additional software in onboard systems and/or HUMS computers often introduce flight safety and validation concerns; the aerospace community is extremely reluctant to introduction of new hardware and/or software in flight approved systems. This has been a significant hindrance to the acceptance of embedded diagnosis solutions based on TEAMS-RT.

The International Space station, for example, is sensor rich. It, as well as most other NASA space systems, transmits voluminous amounts of sensor data to ground support systems (at NASA-Johnson Space Center, Houston, Texas) for health assessment. This data stream is near real-time, and consists of detailed sensor data from multiple subsystems on board the spacecraft. This presents an unique opportunity. We can demonstrate a real-time remote monitoring solution that utilizes this telemetry data to monitor the health of the various subsystems and we can demonstrate the benefits of an onboard solution, without having to actually install any software on the space station itself !

Moreover, sensor-rich systems, lacking a built-in health monitoring capability, are not rare. For example, OTIS has elevator systems, and Pitney-Bowes has copy machines, which are capable transmitting sensor data to remote service centers. Even the modern automobile has plenty of sensors, and some models even have wireless communication links (e.g., OnStar™ systems). All of these systems could benefit from a tele-diagnosis capability, where remote data-streams from multiple subsystems are processed by offsite reasoners for diagnosis and prognosis of the remote system(s). Implementation of such a tele-diagnosis server system is significantly cheaper than embedding reasoners in the subsystems, and would pay for itself by the savings realized from reduction of service calls by technicians.

In fact, in an increasingly connected world, it is not hard to imagine systems ranging from household appliances to sophisticated aircraft systems routinely connecting to remote reasoning services for periodic health assessment and diagnosis. According to more radical views of the future, all our appliances, from the toaster oven to the dishwasher, refrigerators and washing machines, will be networked and inter-operating utilizing technologies such as Jini (a technology invented by SUN) or Universal Plug and Play (Microsoft's answer to Jini). Such a connected world would open doors to a huge market for RDS, with applications in monitoring and diagnosis of millions of appliances.

The RDS Framework

The RDS framework is a three tier scalable server architecture consisting of a middle layer (called "broker") that performs session management, flow control, message buffering and routing, and load balancing. The back ends are server products based on TEAMS-RT (for real-time diagnosis) and TEAMATE (for interactive diagnosis) and a database backend built around TEAMS-KB. The reasoning is model driven, utilizing multisignal models developed in TEAMS. Thus, the RDS framework makes QSI Integrated Diagnostic Toolset [7] (consisting of TEAMS, TEAMATE, TEAMS-RT and TEAMS-KB) accessible over the

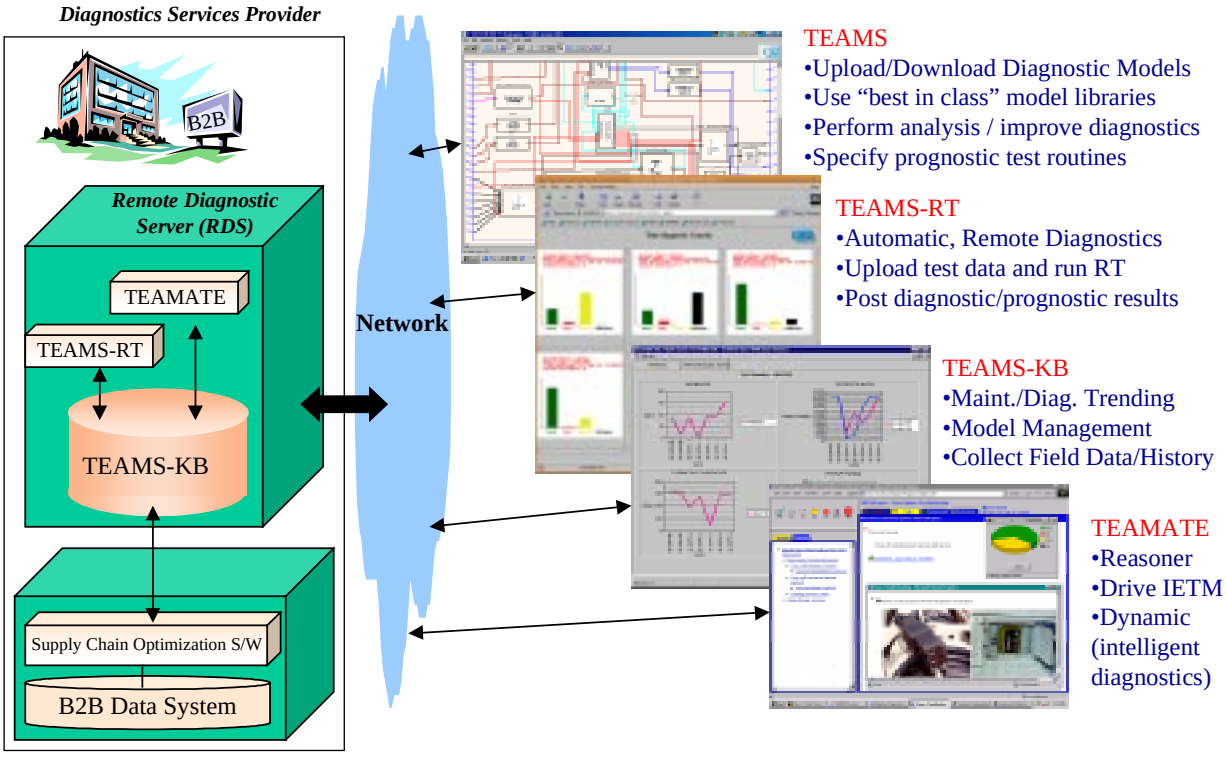

Figure 1: QSI's Integrated Toolset and Remote Diagnosis Server

network to any communication capable system in need of diagnosis (Fig. 1).

Architecture

The essential constructs of the RDS framework had been described in an earlier paper [8] and is not repeated here. A demonstration of remote diagnosis along with a PDF version of the paper is available in http://www.teamqsi.com/rds.Since then, the scope of the architecture has significantly expanded. The major addition to the architecture is the inclusion of a TEAMATE and TEAMS-KB(Fig. 2). The TEAMATE server is based on QSI's TEAMATE tool, which is an adaptive, intelligent diagnostic engine for field (offline) maintenance. Addition of the TEAMATE server allows for a truly distributed, web-based adaptive and interactive diagnosis. Such a capability is of great value to the field technician as it lets him conduct the diagnosis on a local system despite the absence of locally available Interactive Electronic Technical Manuals (IETMs) and models that fit the configuration of the specific system that is being diagnosed. The RDS framework also allows the updates of configuration data and IETMs at the server location that can be safe and secure as opposed to a location in the field. This also results in significant savings in the cost of maintenance and distribution of technical manuals and resources.

The architecture, in keeping with its distributed nature, has been modified such that the intelligent diagnostic engines and their corresponding agents, that allow the communication with the broker, no longer exist as a single entity (Fig. 2). Breaking up the diagnostic engine and the agent allows the two to be distributed across the network. In order to serve a new request, the broker simply starts another agent. Based on the data received from the broker, the agent is responsible to bind to the server for the appropriate diagnostic engine. This allows the broker to function completely independent of the data and focus on tasks that improve the scalability, robustness and the integrity of the architecture. Separation of the agent and the diagnostic engine also allows the diagnostic engines to be hosted on completely separate machines. The

diagnostic state of each of the systems is also maintained as data in the shared memory maintained by the broker. In case of a failure of the diagnostic engine or the machine on which it resides, the diagnostic state can be easily transferred to another such engine residing on a completely separate location. All that is transparent to the user as the agent associated with the session in progress handles the transfer mechanism.

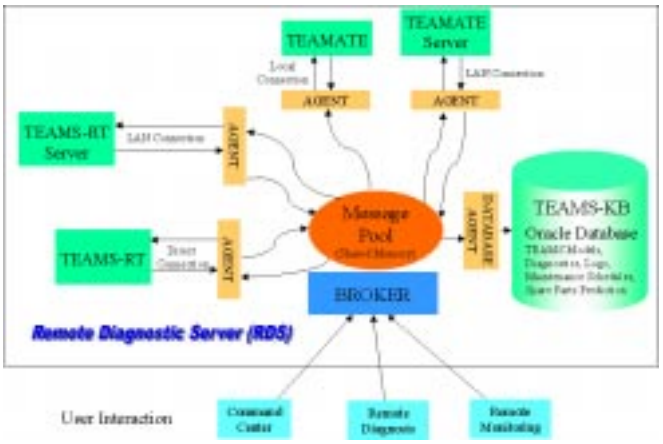

Figure 2: Distributed architecture of Remote Diagnosis within the RDS framework

TEAMS-KB, QSI's database-centered diagnostic knowledge management tool is in the process of being integrated in the RDS framework. TEAMS-KB, an Oracle Enterprise database, has model and diagnostic data management as its core features. Model management includes the capability to create and modify models and components that comprise the model. It manages model and test libraries and along with TEAMS, provides an integrated environment for model development. Diagnostic data management includes capturing and managing diagnostic test and session history logs, managing schedules and tracking and predicting parts requirements. Its interface also allows easy ties to existing custom database systems that perform the above maintenance related actions.

The database agent is a specialized version of the generic agent that is used by the diagnostic engines. The database agents, in addition to the usual read and write capabilities to the message pool, can bind to TEAMS-KB and invoke its stored procedures. It can, therefore, query the database to retrieve or insert the session's diagnostic state, test, and parts replacement history. The database agent obtains the information from the message pool maintained by the broker. Storage of the session's diagnostic state and history data allows easy display of such information by any web-based monitoring console which can use the standard Open Database Connectivity (ODBC) protocol to invoke TEAMS-KB and obtain the required data. Since this process is essentially asynchronous with the RDS server, the display process does not impact the general performance of the RDS server. The database handles all the locking and buffering mechanisms normally associated with such asynchronous processes.

At least three different types of usage are anticipated of the RDS framework, namely telediagnosis, remote health monitoring and a command center console (fig. 2,3). All of the three functionality as supported by the current architecture can involve multiple, distributed systems. Telediagnosis [8] involves diagnosis of such a distributed system where sensor data from the distributed systems are processed and via the message pool of the central broker are transferred to different diagnostic agents. The output of the diagnostic modules is conveyed back to the telediagnosis console via the message pool or via the TEAMS-KB database. The remote monitoring functionality is similar to the telediagnosis feature except that it features only the TEAMS-RT agents which, based on the processed sensor data, updates the health status of the distributed systems in near real-time. The command center console (Fig. 3) is a more interactive version of the telediagnosis feature whereby the user can place different commands to the diagnostic agents via the message pool to actively monitor and diagnose one or many of the distributed systems. The command center console subsumes the functionality of the other two usages and provides the maximum flexibility in terms of controlling the functionality of the different diagnostic agents. The command center console allows interactive diagnosis with a TEAMATE agent and can invoke stored procedures within TEAMS-KB to generate comprehensive, global system health reports and analyses of all the different systems. It can also utilize the failure and diagnostic history of different systems and can invoke TEAMS-KB procedures to display failure

predictions thus allowing the maintenance community to prepare for a possible failure before it actually happens.

Figure 3: A view of the Command Center console

Figures 4 and 5 show some of the interaction between QSI tools, primarily TEAMS-RT, TEAMATE and TEAMS-KB. In Figure 4, the focus is on TEAMS-RT and TEAMS-KB interaction while in Figure 5 the focus is on TEAMATE and TEAMS-KB interaction. We anticipate a central role for TEAMS-KB in both processes as well in the entire RDS framework. When processed sensor data from a new sensor agent are sent to the message pool, the broker, based on the RDS configuration, launches a new agent for TEAMS-RT. Alternatively, the database agent logs the new data to TEAMS-KB, triggering TEAMS-KB to launch TEAMS-RT agent to process the data. The TEAMS-RT agent contacts the TEAMS-RT server with a unique key that is assigned by the broker or TEAMS-KB. The TEAMS-RT server loads the model from TEAMS-KB and processes the current data and any other subsequent data for the session retrieved from TEAMS-KB. The TEAMS-RT agent retrieves the health report from the server and writes them to the message pool. The database agent obtains the health report from the message pool and logs the report to TEAMS-KB for further analysis.

The TEAMATE process can be started in two ways. When the health status report, generated by the TEAMS-RT process and logged in TEAMS-KB, has an ambiguity group, stored procedures in TEAMS-KB gets triggered which launches TEAMATE. Alternatively, TEAMATE can be launched manually through the interface of the command center.

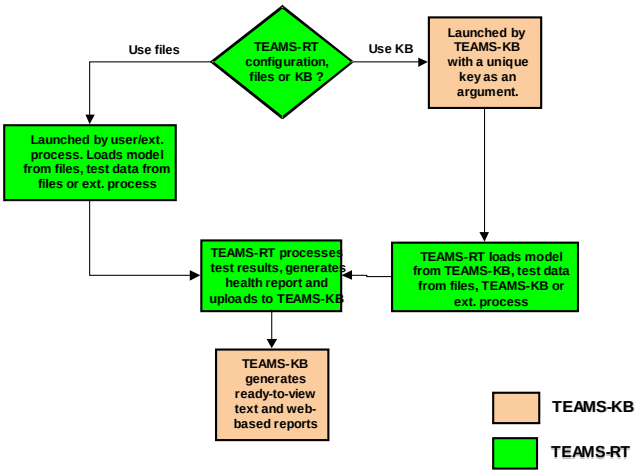

Figure 4: TEAMS-RT processing and its interaction with TEAMS-KB

In the automated launch sequence, TEAMS-KB generates a unique key for the session and launches TEAMATE with the key as an argument. The key also identifies the model to be loaded from TEAMS-KB. TEAMATE loads the model and is ready for the interactive session. Depending on the requirements of the maintenance group, there may be a user login screen for starting the interactive session. Typically, for scheduled and unscheduled maintenance actions, the maintenance group assigns a technician to perform the job. The login process allows tracking the job and monitoring the performance of the technician. The username and password are authenticated in TEAMS-KB and if available, TEAMATE may download the technician's expertise profile from TEAMS-KB. The profile allows TEAMATE to adapt the test strategy and the presentation of the contents from the IETM, to the technician's expertise level.

The interactive session comprises TEAMATE presenting the technician with specific actions, like test procedures to perform. The technician indicates the completion and outcome of the test procedure to TEAMATE. Based on the test outcome, TEAMATE computes the next best test to be performed until the failure is isolated or an

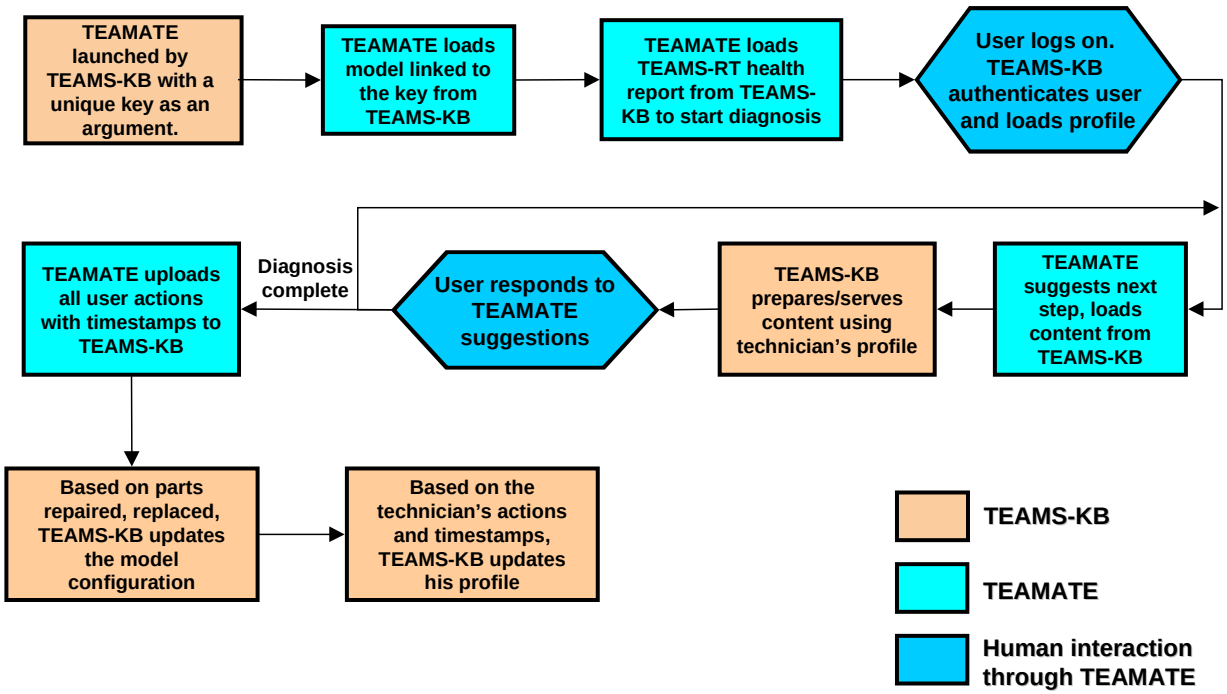

Figure 5:TEAMATE processing and its interactions with the user and TEAMS-KB

ambiguity group is reached which cannot be further isolated. The technician replaces the faulty component/s and goes through the operational check procedure using TEAMATE. At the end of the diagnostic process, TEAMATE logs all the technician's actions as recorded by TEAMATE with timestamps into TEAMS-KB. Based on the session history logged, TEAMS-KB can update the expertise profile of the technician. The parts replacement data captured are used by TEAMS-KB for parts management and prediction.

Simulation Results

We ran our remote diagnosis server, consisting of the Broker, RT-Agent, and TEAMS-RT, on a Sun SS20/502 system with 50 Mhz SuperSPARC processor, 224 Mb of RAM. We picked a slow computer for the remote diagnosis server because the time routines were unreliable in the sub-ms range, and the run-time of the broker could not be reliably measured in the faster computers. We ran the Sensor-Agent clients on both Sun boxes and Windows NT PCs with various hardware configurations. We also ran a web server to serve the monitoring console on the SS20. We collected several performance-related metrics for the different modules and the CPU processing times and memory requirements for the messages on the server-side. We also performed simulations with dozens of concurrent clients to verify the scalability of our architecture. The hardware to host the server, the tests and models that we used were the same as in the evaluation of the tele-diagnosis architecture as reported in the earlier paper [8]. The performance metrices shown here are hence, directly comparable to those of the earlier architecture.

Table 1 shows some of the results obtained for the Broker, RT-Agent and the TEAMS-RT modules for different diagnostic models. The objective here is to determine the computational load to the remote diagnosis server for problems of various sizes. Significant improvement resulted in the CPU times of the RT-agent. Overall, there was an improvement factor in the CPU runtime, of four and a half times in the new architecture. This improvement was primarily a result of the RT-agent reporting results only when the diagnosis changed. The broker and the TEAMS-RT times remained unchanged from the previously reported results. The database agent and the command center console are still in

Table 1: Simulation results for 7 different models of varied complexity

Model	Number of Tests Pass / Fail	Number of Faults inserted / total modeled	RT-agent CPU run time in ms New/old	Broker CPU run time in ms	TEAMS-RT CPU run time in ms
1553	59 / 2	2 / 174	55/250	10	< 5
Transmission system	46 / 5	2 / 160	49/210	9	< 5
EEATCS	9 / 134	2 / 78	55/250	12	< 5
Documatch	175 / 5	2 / 259	60/230	7	< 5
LO2	329 / 39	3 / 167	80/300	7	< 5
Engine System	274 / 32	3 / 255	75/300	7	< 10
LGCU-WRA	1003 / 316	4 / 2080	250/500	12	< 250

development and the runtime performance with these new modules will be reported later. The performance of the agents are expected to improve further as they will no longer be responsible for displaying the results. The command center or the monitoring console which display the results, run asynchronously with the broker and are only limited monitoring console which display the results, run asynchronously with the broker and are only limited by the performance of the database and its throughput.

These results clearly confirm that with the new architecture being more distributed, even the old SS20/502 could scale further and provide remote diagnosis capability to hundred's of clients, as the reasoners are run on remote (faster) computers for models consisting of more than 1000 failure sources (using socket connections between the RT-Agent and TEAMS-RT, as outlined in Fig. 2).

Conclusions

The development of RDS is a major milestone in our plan for commercializing integrated system design, diagnostic and prognostic tools. Our integrated toolset help achieve lower life-cycle costs by addressing reliability, testability and maintainability issues: failure analysis, design for testability, automated testing, interactive diagnosis, and real-time system health monitoring. While many of our competitors offer products in the areas of integrated diagnosis, most lack a real-time diagnosis engine, and none have a networked diagnosis server capability. Until now, real-time diagnosis and prognosis have been available to a selected few multi-million dollar applications. The remarkable aspect of this technology is that it is accessible over internet and modems, making real-time diagnosis universally accessible! This is a key discriminating factor that will enable us to reach beyond the niche market of integrated diagnosis, and tap into consumer applications and e-business.

For example, the modern automobile has enough sensors to detect the slightest performance problem. The engine computer(s) monitor fuel mixture and ignition system for optimal fuel efficiency, drive-train computer(s) monitor the grade of the road, torque and acceleration to select the correct gear, and antilock brake systems detect wheel lock ups and dynamically adjusting for brake wear. Some high-end models already come equipped with communication links (e.g., OnStar™ by Cadillac) that can report mishaps, e.g., an accident causing airbag deployment, to a central monitoring station. In a few years, such features will be available on all cars. Presently, such communication links are offered primarily as a safety net, or as a link to customer and concierge services. However, they can easily be adapted to transmit onboard data to a RDS service where car troubles can be quickly

diagnosed. It is therefore conceivable that soon, the driver of a stalled car will be able to get a prompt diagnosis using RDS service, and AAA would dispatch roadside assistance with the exact spare part required to fix the problem. The applications of RDS are not limited to the automobile. Remote health monitoring of home-care patients and battlefield soldiers are two of the more promising applications. Modern high rise buildings consist of elevators, escalators, heating and ventilation systems etc. that also need to be monitored round the clock. Utilizing RDS, a central facility could monitor entire cities of high-rise buildings from one central location.

RDS is an essential piece of technology that makes such applications feasible.

References

[1] Patterson-Hine, A., Kulkarni, D., Deb, S. and Wang, Y., 1998, *Automated System Checkout to Support Predictive Maintenance for the Reusable Launch Vehicle*, Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, San Diego, October 11-14.

[2] Kulkarni, D., Patterson-Hine, A., Holthaus, M., Deb, S., and Pattipati, K.R., 1995, *Degradation Detection and Testability Analysis in Propulsion Checkout and Control System*, Proceedings of the Aerotech, Los Angeles, CA.

[3] Mathur, A., Deb, S., and Pattipati, K. R., 1998, *Modeling and Real-Time Diagnostics in TEAMS-RT*, Proceedings of the American Control Conference, Philadelphia, June 24-26.

[4] *Joint Advanced Health and Usage Monitoring System - Advanced Concept Technology Demonstration*, Phase I, Final Report, Sikorsky document no. SER 521365, August, 1998. Also, http://www.dt.navy.mil/jahums/ JAHUMS Project Homepage.

[5] *An Onboard Real-time Aircraft Diagnosis and Prognosis System*. Technical Progress Report on NAS2-99048. October 26, 1999.

[6] *A Systematic Integrated Diagnostic Approach to Software Testing*. Technical Progress Report on NAS2-99049, September 27, 1999.

[7] Deb, S., Pattipati, K.R. and Shrestha, R., 1997, *QSI's Integrated Toolset*, Proc. IEEE Autotestcon, Anaheim, CA, pp. 408-421.

[8] Deb, S., Ghoshal, S., Malepati, V.N. and Kleinman, D.L., 2000, *Tele-diagnosis: Remote monitoring of large-scale systems*, Proceedings of the IEEE Aerospace Conference, Big Sky, MT.