

RTOS Selection and Its Impact on Enhancing Time-To-Market and On- Time Design Outcomes

**How time-to-market and on-time design considerations can cost or
make you money - and why you should consider these factors in
choosing or replacing your operating system**

Jerry Krasner, Ph.D., MBA

March 2007

Embedded Market Forecasters

www.embeddedforecast.com

508-881-1850

American Technology International, Inc.

Embedded Market Forecasters
Research and Consulting
for Embedded Products,
Markets and Channels



Executive Summary

New research from Embedded Market Forecasters (EMF) indicates that time to market should be the key selection factor in picking a real time operating system (RTOS). Although in the past RTOS selection was determined by an assessment of the functional characteristics of the candidate systems, the increased complexity and the changes in the embedded market means that these factors are no longer sufficient to provide successful results. With the emphasis on seizing windows of market opportunity, factors which affect time to market have become much more important.

According to EMF survey results, some operating systems consistently outperform others in terms of helping developers get their projects to market on time or even ahead of schedule. This paper examines data that shows the effect of which RTOS was used on whether the project was completed on or ahead of schedule. Since time to market is a significant factor in the profitability of most projects, it's clear that developers should look closely at these figures and make development speed a critical consideration in operating system selection.

Without question, the traditional criteria still matter: the RTOS must also be capable of meeting the functional requirements of the application. Notably, however, the report observes that using an RTOS that is overqualified for the application may have a negative impact on time-to-market. This is because their additional capabilities, beyond what is required by the needs of the application, make these RTOSes more complicated and harder to use.

Introduction

With embedded markets becoming increasingly competitive, the value of RTOS technology has shifted. Developers are confronted by greater design complexities and limited windows of opportunity. Trends indicate that developers no longer have the time to individually research available processors, tools and operating systems; write, compile and debug code; integrate software and hardware and then test the final solution. And, yet, new vertical market application products are emerging weekly, placing increasing demands on embedded developers.

This paper reviews challenges in selecting an RTOS and the importance of the RTOS in achievement of the fastest time to market. After comparing design

outcomes between various RTOS solutions, this paper looks at how selection of an overqualified RTOS affects project timeliness. Problems of delays are examined and practical suggestions of best practices for selecting an RTOS are given.

RTOS selection challenges

Commercial operating systems form a continuum of functionality, performance, and price. These operating systems range from those that offer a basic preemptive scheduler, a few key system services, are usually inexpensive, come with modifiable source code and are royalty free to those more sophisticated operating systems that typically include a lot of functionality beyond the basic scheduler and can be quite expensive. With such a variety of operating systems and features to choose from, it can be difficult to decide which is best for a given project. Many developers make their decision based on performance, functionality, or compatibility with their choice of compiler, debugger, and other development tools. Many use integrated development environments (IDEs) that enable them to develop over a wider range of RTOSes and to use Eclipse-based tools.

When faced with all the traditional selection criteria, choosing an RTOS appears to be a toss-up, with no RTOS standing head-and-shoulders above the others in technical merit. For this reason, RTOS selection has been primarily a matter of preference and convenience. Development teams tend to resist change in the absence of perceived benefits of change. And, since many RTOSes now offer similar technical capabilities and a broad range of software and hardware ports, there are few technically compelling reasons to change.

However, if developers reach beyond the traditional criteria and look at time to market and on-time design characteristics, then some RTOSes clearly rise above others in keeping developers on time and within budget and even more importantly, in getting the product to market faster.

Importance of time to market

Time to market significantly affects the overall cost of a project. Extensive EMF surveys show that the average number of software developers per project is 25.7. If the average cost per engineer (salary, benefits and supporting technical staffing) is only \$150,000 per year, then the average cost for each month of delay is \$321,250.

Knowing that on average 42% of embedded designs are behind schedule by an average of 4.1 months, one can calculate the average expected loss due to design delays per project as $(\$321,250) \cdot .42 \cdot 4.1 = \$553,200$.

Notably, these calculations don't take into account the cost of lost revenues because the product was delivered late. Time to market is important in every industry segment, but in short product lifecycle segments such as consumer products, it often makes the difference between profit and loss.

Clearly, a few tens of thousands of dollars difference in up-front costs to purchase a superior time-to-market RTOS is very small compared to the expenditures experienced as a consequence of late design completion, and the benefits of getting to market ahead of, rather than behind, competitors. Given such dramatic implications, it makes sense that such analysis should become part of every company's best practices.

About the survey

Embedded development engineers were interviewed via a comprehensive survey designed to elicit information regarding current and anticipated tool usage, design starts, completions and cancellations, development (host) and target platforms, microprocessors used, desirable and undesirable product features, vendor evaluation criteria and purchasing decision processes, among other important information. Responses from 510 embedded developers comprised the EMF 2006 comprehensive survey, which explored the attitudes, preferences and values of embedded developers to the current and projected use of embedded technologies. The survey was constructed such that the responses could be evaluated from many perspectives, including total response, specific job title of the respondent, architecture employed in embedded design, processor family, and each of ten embedded vertical market application (e.g., telecom, industrial controls, etc.).

The survey questionnaire was developed jointly by Embedded Market Forecasters (EMF) and Wilson Research Group and programmed for a web-based survey deployment by Wilson Research Group. A base of nth-name selected subscribers to RTC magazine, COTS Journal, and Military and Aerospace Electronics, who indicated on a qualification card a professional area of interest in embedded development, were used as the sample. Starting on 6 March 15, 2006, each member of this sample was sent an email explaining the purpose of the study, that they were statistically selected, and that their participation would help embedded vendors better service the developers. The database was identifiable by a PIN number that respondents used when they went on-line. This served to insure that the appropriate individual responded to the survey and that the response was one-time only. Care was taken to insure that there was no duplication between the two groups. One follow-up email was sent to all non-responding members of the sample two weeks after the original mailing.

Five hundred and ten developers responded to the online survey, of which 55 were hardware engineers, 130 were software engineers, 97 were systems engineers, 52 were systems architects, 43 were firmware engineers, 36 were software engineering or development managers and 35 were hardware engineering managers. In addition 62 developers gave titles other than these listed. This provided an excellent distribution of experiences and viewpoints from which to draw inferences and conclusions. Statistically, the response is at a 95% confidence level, plus or minus 3.6%.

Comparing design outcomes

The EMF survey showed that selection of the operating system has a direct influence on time to market and that by selecting and using the right RTOS, the percent of design completions that are on or ahead of schedule can be maximized. We examined four popular RTOSes to illustrate this data: Integrity (Green Hills), ThreadX (Express Logic), Nucleus (Accelerated Technology), Windows CE (Microsoft), and in-house.

Table 1 presents design outcomes derived from the 2006 EMF survey of embedded developers.

OS	On or ahead of Schedule	Behind Schedule	Months Behind	Months from Start to shipment	New or modified Lines of Code ave x1000	Total Lines of Code ave x1000
Integrity	58.2%	41.8%	4.4	17.6	437	850
Nucleus	61.5%	38.5%	3.9	13.2	85	157
ThreadX	70.5%	29.5%	3.3	13.5	209	699
Windows CE	63.2%	36.8%	3.4	12.2	47	307
In-House	49.3%	50.7%	4.0	14.3	313	1035
Industry average	58.6%	41.4%	4.0	15.5	193	667
For Lines of code <100k	56.7%	43.3%	3.5	12.4	n/a	n/a
For Lines of code > 100k	55.9%	44.1%	4.8	16.5	n/a	n/a
For Lines of code > 500k	53.8%	46.2%	5.5	17.5	n/a	n/a

Table 1: Not all RTOS solutions are equally capable of completing a project on schedule. Compare leading RTOSes to industry averages.

It is clear from Table 1 that developers using some RTOSes complete their designs in a more timely fashion and that even projects behind schedule are completed one month earlier (on average) than those developers using other RTOSes. While there are likely to be applications for which certain operating systems are more project appropriate than others, the data makes it clear that there are compelling time-to-market reasons to use certain RTOSes whenever possible.

It is tempting to assume that this comparison is biased in favor of smaller projects, which might be expected to be completed more quickly than larger projects. Project details, however, do not support this. Figure 1 offers a look at design complexity as measured by lines of code per project.

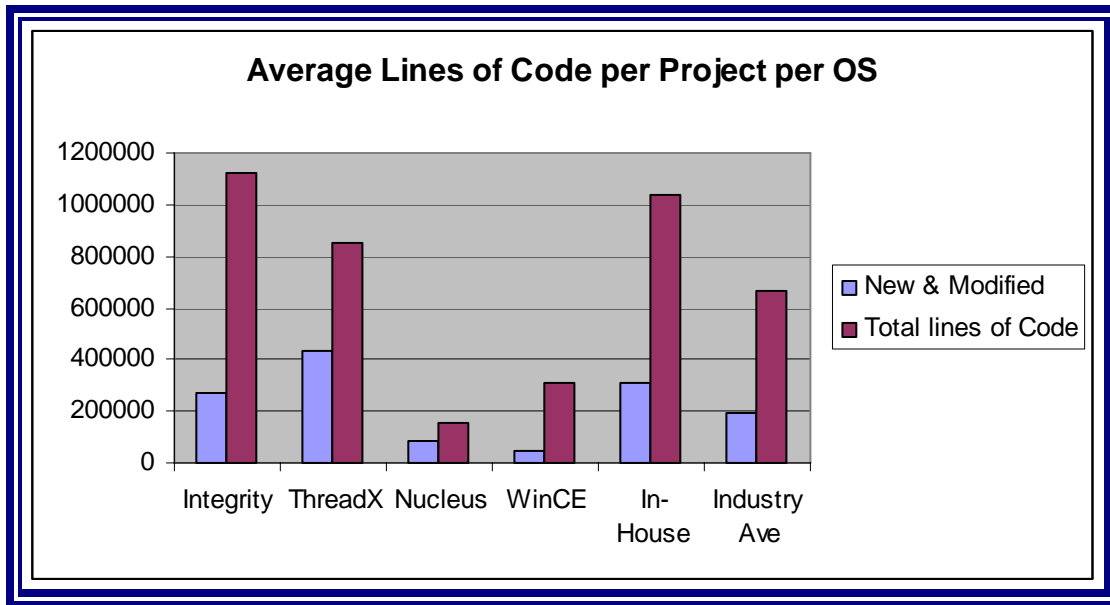


Figure 1: Average Line of Code per Project by OS

Project complexity is often defined in terms of lines of code. Figure 1 shows that there is no correlation between the completion time record of RTOSes and the size of the projects that they are used on.

It is clear from Figure 1 that RTOSes ranking high in fast time-to-market are not limited to projects with fewer lines of code. So the ability of developers to complete projects on or ahead of schedule does not appear to be solely related to project size.

How RTOS selection affects time to market

Why are projects developed using certain RTOSes consistently completed more quickly than projects developed using other RTOSes? The answer may lie in their “ease of use,” a term that is often used to measure the degree to which an RTOS offers simple system services, intuitive naming conventions, good documentation, good support and availability of full source code. These are the characteristics of a product that enables developers to become productive in a short period of time and to complete projects on schedule. Even though ease of use is inherently subjective, there are several factors that are possessed by operating systems that developers describe as “easy to use” and which finish at the top of the chart in terms of projects completed ahead of or on schedule.

One factor that seems to promote ease of use is simple RTOS systems services. This means that each system service provided by an RTOS offers a specific functionality that can be used by the application whenever needed. If the service API is simple, it's more likely to be easily understood by the developer and more likely to be easily learned and used. Just as important, a simple service is more likely to be used correctly, avoiding the introduction of bugs through misuse caused by overly complex services.

Many RTOSes use cryptic, complex names for their services that require a reference guide for the developer to recall what the service actually does. An RTOS with descriptive names for its service calls, organized in a separate namespace apart from application function and variable names, will make development easier, and also will assist in code-review by others who didn't actually write the code.

For example:

(Good): `rtos_message_send(p1,p2,p3)`

(Bad): `xyxConstructBridge_435(p1,p2,p3,p4,p5,p6)`

Avoiding overqualified RTOSes

EMF survey data also indicates that it is important to select an RTOS that suits project requirements. To select an overqualified RTOS results in project delays. For example, the perception exists that "hard real-time" interrupt response requirements are difficult to achieve and that the fastest RTOS should be selected to ensure meeting requirements. This leads developers to demand the highest level of RTOS performance, and thereby leads RTOS vendors to spend lots of time making their RTOS respond faster and faster. Indeed, if such responsiveness were a measure of the benefits of an RTOS, then "more is better."

But, at some point, "more" is not needed. Once response requirements are met, faster response is of little additional value. For example, on a desktop system, it's generally understood that keyboard response of about 10 milliseconds is sufficient to keep up with human typing. A faster response provides no additional benefit, since the human doing the typing won't be able to hit the next character for at least 10 milliseconds.

It's no secret that selecting a processor that is faster than what is required—sometimes called processor overkill—creates design overhead in processor cost,

power requirements, footprint, etc. Likewise, selecting an overly capable RTOS will introduce greater costs in the areas of system overhead, memory footprint, training costs, learning curve, and likelihood of misuse due to its complexity.

Problem of delays in embedded applications

The importance of these concerns is underlined by the fact that embedded designs are notorious for experiencing design delays and cancellations. The problem is expected to become exacerbated as design complexity increases, reliability requirements are made more stringent and security concerns dominate connectivity issues. Tables II-2 and II-3 illustrate the divergence between early and late design completions. Table II-2 represents 2006 data while Table II-3 is 2005 data presented for comparison. It is interesting to note the verticals in which design improvements have been achieved.

Suggested List of Best Practices for Selecting the Most Appropriate OS

1. Calculate monthly project personnel expenses (engineers, managers plus overhead) on a month-by-month basis – this is your cost of project overruns per month.
2. Based on internal past history for similar projects, what was the average number of months behind schedule? Multiply this by the result of Answer 1.
3. What is your forecast revenues based on on-time shipment of product? Realistically estimate the cost of missing the target date (in terms of percentage of market share lost) as a function of the number of months late. For example missing a Christmas delivery schedule might result in the loss of 80% market share and associated revenues.
4. What are the costs of components, power requirement and associated chip requirements resulting from using a familiar but unnecessarily larger and more complex operating system?
5. These are the project “basis” costs which can be compared to the cost of using a different operating system and perhaps new tools sets.
6. What is the cost of development tools, technical support and license fees? What new costs would be incurred by switching OS and/or tools? Unfortunately this is often the only cost consideration calculated by the purchasing agent – and it may be a small fraction of costs incurred from 1, 2, 3 and 4 and totaled in 5.
7. Work with engineering to establish the operating system capabilities and the microprocessor characteristics appropriate to successfully completing the project on-time.
8. Establish a periodic review advisory board to look at advantages and disadvantages of alternative operating systems and tools requirements.
9. Set up a methodology (or subscribe to a service) to benchmark your companies performance against industry and vertical market normals.
10. Don't be afraid of change – assign group responsibilities such that progress can be made and measured without political games or the blame game.

Table II-2: 2006 Percentage of design completions according to schedule

	Ahead	Behind	Cancelled	Outsourced
8-bit	20.2%	42.3%	10.6%	12.9%
16-bit	20.6%	40.5%	10.9%	14.7%
32-bit	18.1%	43.6%	10.7%	14.6%
64-bit	21.7%	41.8%	12.0%	18.2%
DSP	20.7%	41.6%	10.0%	15.9%
FPGA	21.8%	45.1%	10.9%	14.5%
Auto-Transport	27.8%	30.7%	12.6%	14.8%
Avionics	15.6%	54.9%	10.4%	14.0%
Bus Mach & Peripherals	18.4%	42.3%	14.8%	10.4%
Consumer Electr	14.3%	43.6%	13.4%	17.0%
Datacom	11.8%	52.4%	9.6%	6.7%
Telecom	18.5%	38.4%	11.4%	20.3%
Electronic Instrumentation	24.0%	46.4%	5.9%	14.3%
Industrial Automation	23.1%	42.0%	7.4%	9.2%
Medical	20.5%	47.2%	12.3%	14.4%
Military	19.9%	43.3%	9.0%	14.8%

	Ahead	Behind	Cancelled	Outsourced
Auto-Transport	14.8%	55.4%	15.1%	13.1%
Avionics	15.1%	52.0%	11.8%	15.2%
Bus Mach & Peripherals	15.1%	52.9%	14.8%	11.6%
Consumer Electronics	17.2%	52.3%	14.8%	11.9%
Datacom	11.8%	57.2%	16.5%	13.2%
Telecom	10.6%	60.2%	18.3%	7.5%
Electronic Instrumentation	18.3%	57.3%	13.3%	11.1%
Industrial Automation	19.1%	51.3%	13.0%	8.1%
Medical	18.1%	56.2%	11.6%	11.3%
Military	17.6%	52.1%	8.3%	14.3%

Table II-3: 2005 Design Completions by Vertical Market

Enhance your time-to-market outcomes

We have set forth data that shows current design time-to-market outcomes. EMF believes that the data also shows that by carefully analyzing design parameters and requirements *before* the project begins that any OEM or systems integrator can enhance their design prospects and achieve better project completion results by following a series of recommended “best practices” which follows.

Indeed, rather than relying on traditional RTOS selection criteria that have resulted in this declining rate of success, if developers select an OS appropriate to design requirements (avoiding overkill) based on time-to-market, as shown in the survey data, project completion results will be enhanced.

Conclusion

Year-after-year, EMF surveys indicate that developers are creatures of habit and tend to select an RTOS and IDE based on comfort and familiarity. Unfortunately, this tendency appears to result in projects falling behind schedule, and overall development cost increase. Comfort and familiarity can result in project delays, inappropriate scoping of the project requirements and additional design cost. Marketing can come up with the best ideas and the most desirable packaging, but poor design outcomes, unnecessary delays and missed product release dates can be prohibitively expensive and far outweigh any up front cost basis for tool and RTOS selection.

The data presented here can be used as a benchmark for establishing internal controls and insuring design outcomes. It demonstrates that whether or not the project is completed on schedule has a strong correlation to the RTOS that was selected. As a general rule, selecting the simplest and most intuitive RTOS that is capable of meeting the needs of the application will help ensure that project deadlines are met. Time is money and developing an internal program to insure on-time project completions and insure that you meet windows of opportunity in a timely manner will pay huge dividends for your organization.

About the Author

Jerry Krasner, Ph.D., MBA is Vice President and Chief Analyst of Embedded Market Forecasters (a division of American Technology International, Inc.). The company's in-depth surveys of embedded developers are used by industry, the government, and the military and prime contractors to benchmark developer activities, usages and preferences for benchmarking their own processes and for making decisions regarding software, tools and hardware purchases.

Jerry Krasner, Ph.D., MBA
Embedded Market Forecasters
www.embeddedforecast.com
jerry@embeddedforecast.com
508-881-1850

Copyright 2007 by Embedded Market Forecasters, a division of American Technology International, Inc, 1257 Worcester Road #500, Framingham, MA 01701. All rights reserved. No part of this document covered by copyright hereon may be reproduced or copied without expressed permission. Every effort has been made to provide accurate data. To the best of the editor's knowledge, data is reliable and complete, but no warranty is made for this.