Jack Stankovic - I thought I would try a little bit of an analogy talking about what is predictability. We look at, say, a real-time system and what has happened in the past; there are these approaches where the system had to be predictable, for safety critical systems. So people actually built their own special purpose real-time operating system to have complete control over it so they can make it predictable. Then there were lots of demands for using off-the-shelf things, so people tried to use commercial off-the-shelf operating systems like UNIX, and would it be predictable? No, it was terrible, it didn’t really meet any real-time constraint - it was hard to say what the system would do.

And then there’s this conversion that we take a UNIX and make it a real-time UNIX - try to undo the features that are bad for real-time performance and then do we get a degree of predictability? Well, it’s somewhere in between, in fact, you get a lot better than just using off-the-shelf technology, but it still leaves something to be desired, compared to using a special purpose operating system. It is essentially a trade-off that we are giving up some degree of predictability for the benefit of using a commercial kernel that people are used to.
When we look at the database, we might see almost a similar pattern, I think. In real-time databases, if we want predictability and the people have built their own small databases for embedded systems, telecommunications, or air-traffic control, a lot of time these databases are main-memory resident, their transactions are pre-compiled, so they are really trying to be very precise about everything, building it up from scratch. You can get a fairly high degree of predictability there. However, there is a push again for using commercial off-the-shelf systems, like taking a database system and trying to use it, and running on top of a real-time operating system, getting predictability here is pretty much possible. Will this occur? I have not seen this yet, but how about a real-time DB2 database or a real-time Oracle 8 database, or will we start to see the commercial databases trying to adjust what they do to get more real-time properties, so we can get a degree of predictability? This, I do not think has happened yet.

Now, the analogy only goes so far, because of the different situations. In the top part of the real-time system, normally in these systems we know the worst-case execution times, we’re trying to give guarantees. Possibly a 100% guarantee. We don’t want the systems typically that large, they could be pretty large systems but generally not compared to say, large database systems. There is a fault-tolerance requirement that is typically something like "you need to handle these faults in real-time." Many times you are talking about critical applications or at least parts of the application are critical versus in the database world, the problem is not the same, a lot of times here the execution times we really do not know. We do not always have all pre-compiled transactions and even then we do not know the data sets that they are operating with. How important is it? If we do not know this, what are we talking about in predictability? So, here in these kinds of systems, predictability has these underlying issues associated with it. We go to databases, we talk about predictability, but we do not know the execution times, so we are already in trouble. If we want a probabilistic guarantee, or a best-effort behavior, we see that a lot of the metrics are percentages of deadlines, percentage of transactions that make the deadlines or metrics like that. Typically, the databases are large and heterogeneous. There is a degree of fault-tolerance - we want the ACID properties. So if something crashes, the data is still there, it is persistent. If we abort the transaction, however, the effect is as if it did not run. So there is a degree of fault-tolerance.

Generally, if things fail, the system says "Okay, time to reboot, time to recover, and after we recover then we start the system again." That model is not very effective in a lot of real-time systems that this kind of work is performed in. Besides, a lot of times the systems are not critical. We can see this in some of the metrics, such as transaction miss ratio 30% - 40%. What does that mean, what does that have to do with predictability? I think the other issue has to with what if we look at this notion of hard real-time guarantees. They and predictability are related to each other, they are not the same thing, but they are related. If you look at a spectrum of what we had in the past, then if you have a static system designed and off-line, you know all the worst case issues and problem and you analyze it off line. Then you can say that the system is predictable and that you