Learning from Las Vegas

Burton Rosenberg[*]

EXTENDED ABSTRACT

1 Introduction

Our thesis is that the influences that will impinge upon Operating Systems will be so varied, fast changing, and unpredictable, that a substantially new outlook needs to be developed. Very little can be closed down and cast away before beginning a paper design for an Operating System. If one casts aside possibilities, the Operating System will very fast become deficient. We must prepare the foundations so that influences can continually enter and reform the system architecture. Flexibility must be assured at the foundations and intervention mechanisms must be established at the next higher software level. Furthermore, these intervention mechanisms are the tools of a new generation of software architect whose job is to negotiate between the system's possibilities and the end user's desires.

These changes are due to the widespread presence of the Internet and Multimedia. Now the computer is a partner in everyone's daily life, and increasing demands on personal productivity have made this partner indispensable. This has caused a shifting of priorities and created economic imperatives to which research is obliged to respond.

Three aspects are discussed:

1. That the convergence of low and high end computers will make "vernacular" computing a forceful determinate in the future of computing;
2. A new occupation will emerge whose relationship between user and computer scientist is analogous to the traditional architect's relationship between client and engineer. A starting point for the dimensions, techniques and values of this occupation can be derived by analogy from the job of the architect;
3. In the book "Learning from Las Vegas," [1] the virtualization of architecture was proclaimed: "Symbol dominates space. Architecture is not enough." Computer environments were born virtual, but as navigational aids, fictions of solid place and being were taken seriously. Now we are less likely to consider these fictions as helpful. Objects dominates device, and Operating Systems are no longer enough.

These are the ideas I would like to discuss with participants in the workshop. The following sections outline research programs along these lines.

2 The Significance of Vernacular Computing

One aspect of widespread computer literacy is the reversal of flow of invention between research labs and individuals. Fundamental computing technologies still, I believe, move from research lab to the vernacular, but as the success in placing powerful software tools in the hands of individuals unfolds, and as the price of powerful hardware configurations drops, researchers will increasingly be able to look at the "folk language" of computing for inspiration and source materials.

John Ousterhout has clearly verbalized this concern in the specific instance HyperCard:

> The bais idea for Tk arose in respone to Apple's annoucement of HyperCard in the fall of 1987. HyperCard generated tremendous excitement . . . However I was discouraged. The HyperCard system had obviously taken a large development effort,

[1]Department of Mathematics and Computer Science, Univ. of Miami, Coral Gables, Florida. *burt@cs.miami.edu*

> and it seemed unlikely to me that a smal group such as a university research project could ever mount such a massive effort. This suggested that we would not be able to participate in the development of new forms of interactive software in the future [2].

The PDA is a very rich source of new influences. For instance, directory hierarchies are too inconvenient for PDA's, but the alternatives of active search engines already exist. They have arrived spontaneously, and been deployed ad hoc [3].

The point is that the speed of such inventions, and the number of people implementing them will increase, and the economic benefit of tapping this flow is large. This line of inquiry asks: How best can this be done?

3 The Architecting of Computing Environments

The current definition of System Architect focuses on the correct or clever construction of a single software edifice. Once done, it is handed over to System Administrator whose position is more or less subordinate. System Administrators lack the technological leverage by which the excellent System Administrator, as System Architect, can propagate successful solutions to user's demands. An economic dilemma arises: the needs of a single user cannot be met without employing talents incommensurate with a single user's budget.

What is highlighted here is a gap between system builder and system user. The traditional Architect realizes at reasonable cost the integration of user's needs and available technologies. An Architect is considered adequate if he or she solves the problems posed by one client in producing appropriately one building. However, the true art of Architecture is the establishment of systems of solutions from which individual solutions derive. This is the leverage applied in the domain of building-making to make sure that good ideas move quickly out from the particular to the general.

The project here is to consider tools by which a similar leverage can be achieved for the integration of users information needs with available computing technologies.

4 Object Dominates Device

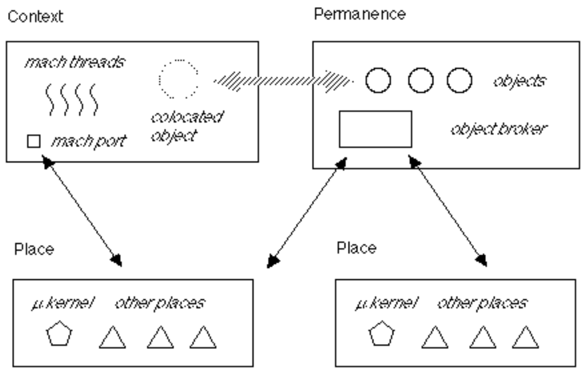
The original Operating System matched jobs to a fixed set of local resources. For efficiency reasons, the Operating System became the embodiment of the local resources. Operating Systems retain their initial goal of efficient resource allocation, but they now must also answer the question of "how-to", not just "how-much". Operating Systems solve efficiency issues surrounding the "implantation" of objects as devices, and require a dialog with networked object brokers to request aid in accomplishing the implantation of a desired object.

The accomplishment and speed of the implantation, e.g. automatically bringing in and configuring a new device, might now outweigh traditional measures as the most important measure of computer performance.

I am working on a microkernel Unix, called Miku, supported in part by software and equipment grants from IBM [4], organized around the three-fold principle of place, permanence and context. *(See figure.)* Places are target abstractions and include the microkernel, device drivers and microkernel services. Permanence corresponds to object resources: amalgamations of data structures and procedures that persist without the constant need for cycles, that self-perserve their data invariants, and can exist simultaneously in several contexts. Context refers to applications: the focus of ad hoc activities with the agendas required to drive objects from epoch to epoch, i.e. effect change to their internal states, according to external needs.

5 Conclusion

We have proposed three concerns for the future of computing, and have claimed that the motor of change is the Internet and Multimedia technology. We have proposed a fresh look at software organization, to be explored in the Miku Operating System. We hope that learning from Miku might be be useful in responding to these concerns.



Miku Structure Principles

References

[1] Robert Venturi, Denise Scott Brown, Steven Izenour. *Learning from Las Vegas : the forgotten symbolism of architectural form.* MIT Press, 1977.

[2] John K. Ousterhout, *Tcl and the Tk Toolkit.* Addison-Wesley, 1994. Page xvii.

[3] Eric Freeman, *Lifestreams for the Newton,* PDA Developers, Vol 3/4, July/August 1995. Page 42.

[4] *IBM Annonunces New Software Code That is Universal,* Wall Street Journal, October 31, 1995. Page B6.