Process Management Reviewed

Mohsen Sharifi[1], Seyedeh Leili Mirtaheri[1], Ehsan Mousavi Khaneghah[1]
[1]School of Computer Engineering
[1] Tehran, Iran
{msharifi, mirtaheri, emousavi}@iust.ac.ir

Zeinolabedin Mosavi Khaneghah [2]
[2]Faculty of Management, University of Tehran
[2] Tehran, Iran
zmousavi@pmamut.com

***Abstract*—We propose a new 5-layered pyramid of process needs that must be administered by local and global managers of the process society, namely the computer operating systems of the future. We also propose a new needs-oriented approach to process management based on the proposed categorized process needs. We argue that in contrast to traditional blind order-oriented operating system process managers, needs-oriented process-aware operating system process managers are more favorable to future computing environments with relatively large-scale orders of magnitudes of processes and resources scattered in smallest and biggest imaginable scales and varieties.**

Keywords-process; process management; process needs; operating systems; process-aware

I. INTRODUCTION

Computing systems have long entered and instrumented human societies in recent decade so much so that most are advocating electronic societies in which computers act on behalf of humans. These actions are thus quite vital to human well being and require meticulous criticism.

We believe in the revival of human understanding of and attitude towards computer entities that actually denote these actions namely *processes*. We advocate a U-turn on our viewpoint on event-driven computer processes, replacing the old black-box view by a more autonomous process-aware view wherein each process has a repetitive though limited life cycle of providing services to categorized discrete requests from outsiders.

We envisage a more reasonable management of locally and globally distributed computer processes than traditionally achievable. In the same way Abraham Maslow [1, 2] has categorized human needs in a pyramidal hierarchy, we categorize and structure process needs in 5 layers. In this hierarchy, self-consciousness is at the top and vital needs at the bottom of the pyramid, and security and social and power needs are in between, for the purpose of better management of societies and their needs [3].

Processes need recognition by the process society to begin with. This implies that each process must be granted a globally unique identifier as a first citizen entity and enough local and global space to start living. Having got the vital resources from local and global administrators and managers of the process society, they require proper means for inter and intra process communications to autonomously and in their own discretion pursue their goals and objectives prescribed at their birth time.

There are no aimless processes in the process society and processes need to communicate and cooperate to achieve their goals in their limited lifetime. In the race for resources, processes may wish to get more privileges from the process society compared to other processes in order to get to their goals. The managers of the society must thus provide some sort of improvising the priorities.

Processes get more self conscious and aware of their own behaviors and their society as time passes and they come closer to their termination time. Although not all processes might be concerned with their security, they are all vigilant on their safety all along their lifetime.

We thus envisage a 5-layered pyramid of process needs that need to be administered by local and global managers of the process society, namely the computer operating systems of the future. We can now draft a new philosophy for the management of processes based on the given categorized needs in terms of lifetime-needs. We argue this philosophy is more favorable than traditional process management of operating systems to future computing environments with relatively unbounded large-scale orders of magnitudes of processes and resources that are scattered in smallest and biggest imaginable scales.

The rest of paper is organized as follows. Section II presents the traditional definition of processes and how they are used to be managed in operating systems. Section III argues in favor of a change in the traditional view and proposes a new set of definitions and roles for processes and process managers. Section IV argues how the propositions can benefit future computing and Section V concludes the paper.

II. TRADITION

Believing in a purely and fully mechanical world, we humans are used to envisage a manufacturing factory as a complex of (electro) mechanical tools operated by a number

 ISBN: 978-1-61208-154-0

of human (and more recently, humanoid) operators to produce as many products as possible at the lowest costs. Executive managers on the factory floor take their orders from top management and monitor and control these orders are exactly followed by the operators to the last point with no regard at all to the consciousness and awareness of operators on what and how they really do the work. The prominent paradigm is purely order-oriented. Operators and executive managers that take orders best irrespective of how they feel about it or whether orders satisfy their requirements are considered as best. This is what used to be followed and advertized by the adverts of Taylor school [4, 7].

The operators and executive floor managers are the active entities, i.e., *processes* and *process managers* respectively, in this game. Those who have originally designed the factory, i.e., *creators*, though humans but are not active in the runtime manufacturing process and are not willing to sacrifice production rate by any unanticipated runtime requirements of operators or executive floor managers. Even worth, they do not grant much authority to executive floor managers to manage operator society differently than already prescribed at design time.

This Taylor style of management only works fine in cases where everything is known correctly and completely (i.e., accurately and exactly) a priori and proper design is sought based on this valid and complete knowledge, does not requiring to bother about any unforeseeable changes at all. This condition had never been satisfied in real world though especially in a working set including humans as their principal active entities of work.

We have experienced a similar dogmatism in the management style of computer operating systems too. Problem domain experts eager to use computer were forced to think they know the exact solution to their exactly known problems. Programmers were forced to think the advocated solutions by expert domains were exact, which had to be followed in every prescribed step and sought states. Operating systems followed suit and supposed that programmers exactly knew the constraints of operating systems and the hardware they ran on. They thus executed *processes* running these programmed solutions in the exact orderly-prescribed manner. This worked fine as long as no change was made to any suppositions. The executive floor manager (i.e., *process manger* of operating system) controlled the exact orderly-prescribed execution of processes without any regards for any possible process needs. This is to say that it could have denied the basic needs of a process but still expect the orderly execution of the process.

We do not live in utopia anymore [2, 9]. We are currently living in an information era with very unknowns but yet many hard requirements to be able to work and service irrespective of our many unknowns. Nowadays, domain experts no more claim to have the exact knowledge about their problems and solutions to these problems a priori. Programmers are no longer rest assures that they have programmed the exact solutions to problems. Processes expect to face unforeseen cases not orderly-programmed before. Process managers have to manage ever-increasing resource hungry processes that race for additional new types of needs too (e.g., membership, safety, security). Process communities are getting worldwide and wider under disparate communities. New varieties of resources scattered and distributed in different administrative domains have emerged. Voluminous data is generated as time passes which needs to be fed to processes at runtime with many runtime constraints. At last but not the least, requirements are changing fast in all dimensions and at all levels. In short, we live in a dynamic compute world.

This new compute world requires a completely different approach and style to management of its constituent active entities, i.e., *creators*, *process managers*, and *processes*. It is no more possible for designers to prescribe orderly-prescribed execution of processes and expect process managers to handle all changes in process needs by themselves.

In the next section, we present a purely managerial solution to the general problem and then present our proposition accordingly for operating system *process managers*, *processes*, and *creators.*

III. PROPOSITIONS

Maslow has presented a new more conscious management style that is more akin to work in an ever-changing real world. He has rightly replaced the old purely *order-oriented* approach with a *needs-oriented* approach to management, categorized in a five level hierarchical pyramid (Fig. 1) [5, 6]:

1. ***Physiological Needs*:** for food, drink, air, sleep-the basic bodily "tissue" requirements.
2. ***Safety Needs*:** for security, stability, protection from harm or injury; need for structure, orderliness, law, predictability; freedom from fear and chaos.
3. ***Belongingness and Love Needs*:** for abiding devotion and warm affection with spouse, children, parents, and close friends; need to feel a part of social groups; need for acceptance and approval.
4. ***Esteem Needs*:** for self-esteem based on achievement, mastery, competence, confidence, freedom, independence; desire for esteem of others (reputation, prestige, recognition, status).
5. ***Self-Actualization Needs*:** for self-fulfillment, actually to become what one potentially can be; desire to actualize ones capabilities; being true to ones essential nature; be what one can be.

In short, the Maslow school of management believes that humans (a la *processes*) can be expected to act more responsibly and effectively towards community goals only if their evolutionary declared needs are recognized in their societies (*creators*, *process managers*, and *other processes*) and are incrementally satisfied by their communities in their lifetime.

 ISBN: 978-1-61208-154-0

Self Actualization Needs
Esteem Needs
Belongingness and Love Needs
Safety Needs
Physiological Needs

Figure 1. Maslow human needs hierarchy [3, 8].

Taking Maslow's managerial approach as a base, we have envisaged and propose a 5-layered hierarchical pyramid called Melz-Mazlow pyramid for compute process needs in the compute world of today too (Fig. 2), whereby processes are considered as the smallest active entities executing solutions to domain experts' problems (operating system processes are excluded):

1. ***Existential Needs*****:** for storage space (memory, cache, disk, and file), and processing time (CPU, GPU, and Core) – the "vital" basic requirements.
2. ***Safety Needs*****:** for process security, stability, protection from harm, attacks, and injuries; need for structure, orderliness, law, predictability; freedom from fear and chaos.
3. ***Openness and Belongingness Needs*****:** for process communication mechanisms (IPC, DSM, MP) to converse with child processes, parent processes, process managers, and other processes outside local domains; need to become a member of and communicate with other compute processes and social groups; need for acceptance and approval.
4. ***Competitiveness Needs*****:** for process self-esteem and survival based on race conditions, achievements, mastery, competence, confidence, freedom, independence; desire for esteem and referential position amongst other processes (priority, leadership, reputation, prestige, recognition, status) and desire for the good of other processes (binding, cooperation, trust, friendship, richness, upgrade, in addition to their reputation, prestige, recognition, and status) too;
5. ***Transcendence Needs***: for process self-awareness, self-fulfillment, actually to become what a process can potentially be and is expected to achieve at its limits; desire of process to actualize fully its capacities and capabilities; being true to ones essential nature; be what a process can be – the ultimate "*behavioral* self-consciousness" requirements.

Transcendence Needs
Competitiveness Needs
Openness and Belongingness Needs
Safety Needs
Vital Needs

Figure 2. Proposed Melz-Mazlow compute process needs hierarchy.

Using the proposed Melz-Mazlow 5-layered process needs, we now propose a request-based approach in place of an order-based approach to process management in operating systems of the future.

To begin with, it is essential to reiterate and remind ourselves about some of the important best practices in operating systems design in the past that lay the best grounds for design and implementation of any future operating system, including its process management:

1. Operating systems need not get involved in policies but rather should take the policies as input and try to enforce them as best as required by making the most intelligent use of compute resources; process managers must follow suit and avoid from getting involved in deciding on policies on behalf of processes.
2. Operating system kernels must be kept as primitive as possible just to satisfy the bare requirements not all requirements.
3. Operating system designers and developers are not engaged during execution of processes explicitly. However, they could implicitly influence the executions of processes by embedding runtime mechanisms to enforce their intensions. In other words, *creators* of operating systems are not engaged in day-to-day operations of processes. Operating systems act on behalf of their creators.
4. Process managers of operating systems are synonymous to factory floor executive managers that manage operating processes running solutions to domain expert problems.
5. Operating systems can perform more effectively to satisfy the requirements of processes if they are made aware of the overall requirements of processes well in advance of executions of processes either statically or dynamically but with the least overhead on process management and total performance of the system.

The question is now how does a process manager that is intended to work based on process needs differ from traditional process managers that take orders from their creators blindly with no regard for process needs. The answer is simple: process manager must be reflective to process needs.

Information on each process are stored in kernel structures as before but just an snapshot of these information is given to the process upon every request of process so that it can look at these information and get informed about its status quo so that to guide the process manger what to do to best satisfy its needs. This way the process knows by itself and takes the responsibility that it is not reasonable to ask for its safety needs if it had not asked for its vital needs yet or its previous calls for vial resources had not been satisfied yet. In turn, it does not engage itself in communication with other processes if it has not asked for protection yet or that it had asked for it before but not provided yet by the process manager or other managers concerned. Only an open process enabled to communicate asks for socialization with other

 ISBN: 978-1-61208-154-0

processes to strengthen its competitive edge in its own local society or in other global societies. Finally, only a process that is conscious about its position in communities will be willing to cater and ask for its transcendence needs.

Therefore, what we propose here is that processes behave more wisely and orderly by themselves and allow the process manager to take their more thoughtful needs and manage them all more wisely as guided by processes. This is quite in contrast to traditional processes that go wild to ask for as much as possible of everything irrespective of their status quo, burdening the process manager with how to resolve many possibly conflicting requirements and to satisfy the blindly requested needs of all processes.

The gaining of the process manager is tremendous at the cost of merely providing the current image of information it has on a process to the process upon handling of every request of the process. The trick is that process manager becomes process aware whilst it is still kept primitive with the least thickening of the operating system kernel.

Interestingly, such a reflective process manager is expected to be scalable to handle global distribution and higher varieties and numbers of process needs.

IV. ARGUMENT

Let us now briefly argue how our propositions can benefit future computing. We noted that tomorrow's compute world is faced with numerous new applications that try to solve complex problems whose behaviors might not be fully understood and coded statically. Such applications require the means that can detect these behaviors dynamically. For example, MM5 [8] is a well-known traditional weather forecasting model that uses a very restricted number of weather parameters such as temperature and humidity, and takes the integral of a formulation of these parameters to predict weather in a given meshed geographical region. To improve the accuracy and geographical extent of predictions, more advanced models such as the WRF model [9] have been introduced. Interestingly, WRF uses the same algorithms as in MM5 for small size regions, a la taking the integral of a number of weather parameters though larger in numbers than in traditional MM5, but unexpectedly changes its behavior by taking derivations of parameters when forecasting weather for inter-regions that may be due to unexpected changes in the wave structure and energy. This is to say that the behaviors of processes comprising such an application are at the least very complex and hard to predict statically, implying that the pattern of their requests for compute resources are also hard to predict.

The old order-oriented approaches to management of such processes take one of the following actions in response to requests of a process for compute resources (i.e., CPU, I/O, file, and memory):

1. Blindly satisfy or reject the request of each process solely based on the local availability or unavailability of the requested resource irrespective of previous pattern of resource requests of this process and irrespective of the role and relation of this process to other processes in the community of this process that together make up the process population of an application. We believe this kind of ah-hoc response by process manager to process requests cannot satisfy many quality requirements of future computing applications such as high performance requirement.
2. Decide how to handle the request based on compile-time analysis of process requests. This is restricted to applications whose resource request patterns of processes of an application can be fully determined at compile-time; the very condition that is hardly satisfied in future complex applications such as the stated WRF weather forecasting example.
3. Deploy an additional component in the operating system (process manager) to profile the pattern of resource requests by the process through repetitive runs of the application to which the process belongs, and use this pattern to handle the request accordingly. Apart from the high run-time overhead of profiling, the addition of an extra component in the operating system for this purpose contradicts with the principle of keeping the operating system as primitive as possible. Furthermore, as in the first ad-hoc alternative, the decision on how to handle the request of the process is made irrespective of the role and relation of this process to other processes in its community that together make up the process population of an application.

The proposed needs-oriented approach to management of processes comprising a complex application uses the proposed category of process needs to avoid the shortcomings of the traditional order-oriented approaches. It achieves this by deciding on how to handle requests of a process based on current needs status of the process at run-time considering the patterns of resource requests of all other processes in the community of this process together forming an application. It does not require repetitive runs of application and saves us from adding a profiler to the operating system too. Nevertheless, how can this be done?

Without entering into implementation details, we suffice to answer this question by presenting our first intuition on how we may implement this kind of process management.

We know that the operating system keeps all types of information on processes in its own data structures in the kernel space. The process manager can share this information on a process with the process before deciding how to handle the request of the process. This reflection allows the process to request more consciously according to its hierarchy of needs (as proposed before). To give an example, a process that finds out that its vital needs have not been satisfied yet, stops asking for transcendence needs in vein.

On the other hand, the process manager can look up at current values in data structures storing information on all processes of an application and find out about the needs states of processes to determine if it had to upgrade the level of needs status of any process or not. It can also get a clue on future resource-request patterns of processes and accordingly

 ISBN: 978-1-61208-154-0

reconfigure itself dynamically. For example, it can kill the random identity generator process that generates unique system-wide identifiers for new forked processes in the application if it expects that the application will not fork new processes anymore. It may start working out how to deploy a remote resource if it finds out that a process will ask for the resource soon but it knows that this resource is not available locally. Overall, processes submit requests less frequently more sensibly and more orderly, process manager is more aware of request patterns and needs status of processes allowing it to be more responsive and respective to reasonable requests of processes representing the community of an application at run-time. This all leads to a much better well structured and well behaved process society that can handle the complex and extremely dynamic applications of the future.

V. CONCLUSION

Arguing against the shortcomings of traditional process managers for proper handling of process requests, and by using the Maslow's school of human management, we proposed a 5-layered pyramid of process needs that local and global managers of the process society, namely the computer operating systems of the future, must administer. We also drafted the responsibility of process managers based on the given categorized needs. We think conscious processes alongside process-aware process managers of the kind presented in this paper build up a more promising and manageable computing environment that has relatively large-scale orders of magnitudes of processes and resources distributed in wide scales.

We are currently working on a prototyped implementation of a Linux-based process manager enhanced with the proposed mechanism in this paper to provide process consciousness in terms of categorized process needs.

REFERENCES

[1] A. H. Maslow, Maslow on Management, Wiley, Ed.1, 1998.

[2] A. Adler, Social Interest, Faber & Faber, London, 1938.

[3] D. O'Connor and L. Yballe, "Maslow revisited: constructing a road map of human nature", *Journal of Management Education*, vol. 31, no. 6, pp. 738-756, 2007.

[4] P. F. Drucker, Management Challenges for the 21st Century, Butterworth-Heinemann, UK, 2006.

[5] E. L. Deci and R. M. Ryan, "The what and why of goal pursuits: human needs and the self-determination of behavior", *Psychological Inquiry*, vol. 11, no. 4, Page 227, 2000.

[6] J. J. O'Toole and K. J. Meier, "The human side of public organizations contributions to organizational performance laurence", *The American Review of Public Administration*, vol. 39, no. 5, pp. 499-518, 2009.

[7] P. Lawrence and N. Nohria, Driven: How Human Nature Shapes Organizations, Harvard Business School Working Knowledge, 2001.

[8] Ó. Rögnvaldsson, J. W. Bao, H. Ágústsson1 and H. Ólafsson, "Downslope windstorm in Iceland – WRF/MM5 model comparison", *Atmos. Chem. Phys.*, vol. 11, pp. 103–120, 2011.

[9] R. O. Olatinwo, T. Prabha1, J. O. Paz1, D. G. Riley and G. Hoogenboom, "The Weather Research and Forecasting (WRF) model: application in prediction of TSWV-Vectors Populations", *Journal of Applied Entomology*, vol. 135, no. 1-2, pp. 81–90, 2011

 ISBN: 978-1-61208-154-0