



Message from the Chair – January, 2012

Allen Flynn, PharmD, CPHIMS, CHS

Manager, Medication Use Systems Team, University of Michigan Health System
Chair, Section of Pharmacy Informatics and Technology (2011-2012)

sections@ashp.org

Greetings and salutations,

Happy New Year 2012 (11111011100)!

Since last July, this monthly message has focused mostly on software, including specific topics such the need for precision when describing software, the importance of information representation design, and our call to humanize information technology, in part by improving medication-related software.

I previously argued that *software matters*¹, whether we define it as a list of instructions that can be executed by a computer or, more grandly, as a set of *conditions of possibility* governing all computer use.

At the outset of 2012, let's examine the regulating nature of software a little BIT more.

We can begin by looking for themes within the definitions of the following words; words that are all related to software:

Requirement, n., something obligatory that is desired or needed

Instruction, n., an authoritative direction to be obeyed, including a coded command to the computer

Algorithm, n., a finite set of unambiguous instructions performed in a sequence to achieve a goal

Code, n., a system of symbols and rules used to represent instructions to a computer

Application, n., a computer program with an interface enabling people to use the computer as a tool

¹ [Chair's Message, July 2011](#).

Notice the theme of *control and authority* in these computer-oriented **definitions**. Programmers and users both exert control, but in very different ways. The software development process begins with *obligatory* end-user **requirements**. Programmers must transform end-user **requirements** into *commands* that the computer will *obey*. An application is developed by aggregating *commands* into **instructions** and **algorithms** which, when combined, form fixed **code**. Users run **applications** to *direct* their activities while using the computer. Finally, when *directed*, applications generate *constrained* outputs intended to satisfy previously shared **requirements**.

Noting that all code, and therefore all computer output, *must always* be associated with constraints is important. Lawrence Lessig writes that software and hardware together “constitute a set of constraints on how you can behave.”² We are, Lessig claims, partly regulated by our software along with other forceful regulators, namely, (a) laws, (b) norms and (c) markets.³ The full implications of software as a forceful regulator governing the profession of pharmacy, indeed governing all of the health professions, may not be fully appreciated.

Next, consider how, as code becomes application, and as application provides a set of constrained outputs to the user, interdependent *patterns* of computational activity and human behavior result. Think of the early video game, Pong.⁴ Pong constrained the player (user) by restricting the virtual “paddles” only to vertical movements. Players responded to this software constraint by moving the paddles up and down to keep the virtual “ball” from leaving the video monitor and prevent their opponent from scoring a point. A *pattern* of interactive game play resulted.

Suber proposes that software is “pattern *per se*, or syntactical form.”⁵ Within his essay on the metaphysical nature of software, Suber pragmatically acknowledges that “from the top down, software is pattern we use to do work.” Perhaps, applying Suber’s insights, we can define software in yet another way, **as a unification of pattern with purpose**.

Allowing that software is **pattern with purpose**, I conclude that the software we use in professional practice both reflects and establishes many intended pharmacy practice patterns, while it helps regulate pharmacy practice. Therefore, if we wish to change the patterns of pharmacy practice to meet the goals of ASHP’s [Pharmacy Practice Model Initiative \(PPMI\)](#), we must change the software that pharmacists use by incorporating the new practice requirements.

² Lessig, Lawrence. [Code version 2.0](#), Basic Books, 2006, p. 124.

³ Ibid, p. 123.

⁴ See <http://en.wikipedia.org/wiki/Pong>

⁵ Suber, Peter, What is Software?, accessed January 8, 2012 at: <http://www.earlham.edu/~peters/writing/software.htm>

In the information age, transformation means reforming purposeful patterns by writing and adopting new code.

We require a transformed pharmacy practice that includes routine medication history taking, medication reconciliation, indication tracking, medication-use planning, documentation of effects and outcomes, individualized patient education and longitudinal medication therapy management. Our practice model goals need to be restated as obligatory professional requirements that programmers can render as code, providing pharmacy with new and updated applications for the intended patient-side, relational, transformed pharmacy practice.

My premise that software matters, and greatly so, is now fully explained. Next month we will examine a related, but different topic, workflow.

Write me at aflynn@healthpracticeinnovators.com and share your thoughts and ideas on how we can wisely generate requirements to update the software used by pharmacists so that the pharmacy practice model is positively transformed.

Best Regards,

Allen Flynn, PharmD, CPHIMS, CHS

Chair, Section of Pharmacy Informatics & Technology (2011-2012)

Solutions Designer

Health Practice Innovators