Task Programming Methodology for Powered Wheelchairs

Giles E Tewkesbury, David A Sanders, Malik Jamal Musa Haddad,
Nils Bausch, Alexander Gegov, Ogechukwu Okono
University of Portsmouth
giles.tewkesbury@port.ac.uk

Abstract. The work described in this paper is directed towards applying task oriented methodologies to the creation of new types of powered wheelchairs. The new work has required the creation of a new type of task machine. In addition, new techniques to analyse situations and implement the results within a new user interface have been investigated. This paper describes task machinery in general and the stages required to create a task machine.

Keywords: Wheelchair, powered, task machine, collision avoidance, design.

1 Introduction

The concept of a task machine was first described by Strickland [1] and then extended to apply it to design using advanced production machinery such as 6 degree of freedom robots [2]. Task machines are machines built or constrained specifically for a task and that are not product dependent. A task machine would therefore be defined in terms of a task rather than functional abilities; for example, "an expert machine whose only function is to spray paint panels", rather than "a 6 degree of freedom robot". The task-oriented approach suggests that rather than pursuing single robot structures which can ‘do everything', a task robot should be tailored to the production task and designed to the permanent aspects of that task.

Three directives which outlined the task-oriented approach were [2]:

- Directive 1: Design industrial robots for the permanent aspects of the surrounding production environment, and the task at hand.
- Directive 2: The robot controller should be expandable to the requirements of the customer, from a set of generic modules and the robot itself should be constructed from modular robotic elements.
- Directive 3: Robots should be task, not functionally, programmed using _native' production languages or graphical interfaces.

Figure 1 shows the development process for a task-machine. The definitions, requirements and specifications of the task are the starting point for the machine. The specifications should encompass all the variations that would ever occur within the task being performed, i.e. they should not be product dependent. These are then used in the design of both the software [3] and the hardware for the task machine. The hardware would either be purpose built or constructed from modular elements.

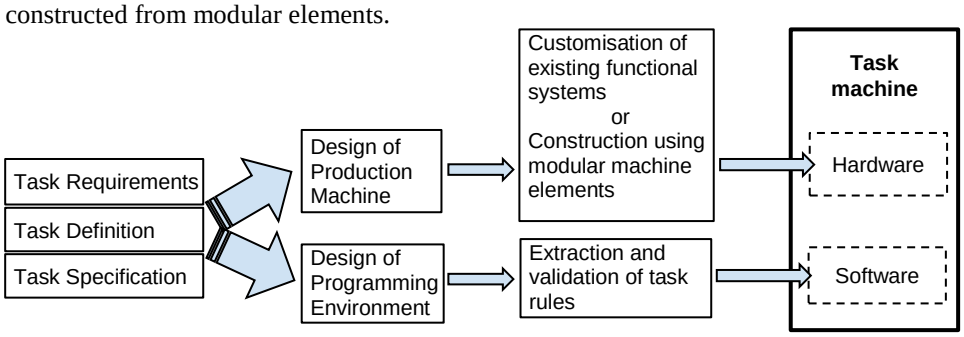

Fig. 1. The Creation of a Task Machine.

Advanced wheelchairs [4-10] can be viewed in a similar way to advanced production machinery, in that they are a fusion of actuators sensors and control systems [11-15].

Wheelchair systems are constructed from modular elements which may include motors, sensors, and user input [16-22]. The needs of each user differ, and therefore may require different user interface devices [23-31]. Langer discussed a range of sensors that can be added to wheelchairs for detection of walls and objects [32].

Within this research, the task machines used either switch controls, or proportional input joystick controls [10]. The types of sensors available were a scanning collision avoidance device (SCAD) [32] using ultrasonics [9,11,12], mechanical bumpers and infrared proximity detectors [21]. Figure 2 shows the prototype wheelchair base fitted with the SCAD detection system [32], with two types of input device (joystick and switches[10]).

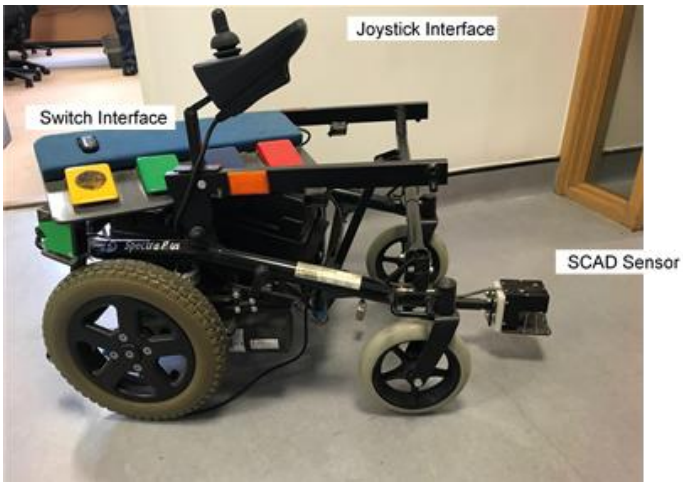

Fig. 2. Prototype Task Machine with SCAD sensor.

New techniques to analyse user requirements and implement the results within a new user interface have been investigated.

The review of literature by Tewkesbury [2] covered several subject areas, including analysis techniques, hierarchies for programming, programming languages and design. It also describes the creation of virtual task-machines and task-machines. The review of programming methods for machines suggests that most systems have been designed to be programmed by experienced programmers. That makes the creation of systems tailored to individual user requirements expensive and difficult. Many authors have proposed methodologies that may lead to the creation of systems which can be reprogrammed by users with little or no programming experience. These methodologies divide programming concepts into levels of sophistication, or methods of control.

The approach used for the work described in this paper has been to focus the programming of a system with respect to the users understanding.

To avoid any inconsistency between the end users and the system experts, it is necessary to define the roles of the programmer and the end user. The user will know exactly which tasks needs to be performed, but have no knowledge of programming. The terminology and concepts a user would be familiar with would differ from those a programmer would be familiar with, and may differ from one task to another. it is worth noting that there may be multiple end users, for instance a clinician may be regarded as the end user rather than the wheelchair operator in some cases, as the clinician may have specific goals that they want to achieve with the wheelchair operator. For example, providing the freedom/opportunity for a user to crash the wheelchair into walls whilst providing a safe velocity and environment to achieve this.

The approach presented in this paper builds on the principal that a user should only program using familiar concepts and terminology, and that lower level programming would be performed by an experienced programmer. The recognition that users should not program a machine is a first step towards creating task machines.

2 Task Machinery

Various authors describe the concept of a `task' in multiple ways. In this paper a `task' is specified as a single specific operation, such as navigating from one location to another. Subtasks may include for example, object avoidance [33,34], doorway navigation [5], speed control and terrain navigation [35,36].

A task machine has been defined as a machine that is knowledgeable in the area of one task [2].

Tewkesbury detailed some of the work in programming task machinery [2]. There are currently three types of task machine: a true task-machine, a surrogate task-machine, and a virtual task-machine. The end result is always a machine that is constrained to a task. These three types of machine can be classified as follows:

A true task-machine is a machine which has been developed specifically for a task. The requirements of the task are used to define the mechanical structure of a machine. For example, if a task were being automated that only requires motion in an X-Y plane, then a two-degree of freedom structure would be created. Strickland discussed this type of task machine [1]. A surrogate task-machine is a commercially available machine that has been selected specifically for a task. The physical structure of a task machine can be selected from the general-purpose machinery available, but may not be entirely suited to the task. To create a surrogate task machine the controller is replaced by a distributed controller, which constrains the machine for a specific task. A virtual task-machine is a general-purpose machine, which has been constrained in software for a task. The structure of the machinery may not be entirely suited to the task and some of the functions available within the controller may be redundant, however, this type of task machine allows existing production machinery to be utilised within the task approach. This new type of task machine is described in full in [2].

The new work described in this paper includes the creation of two virtual task-machines. These machines were based on two general purpose powered wheelchairs, where the generic controllers are dynamically constrained for thee specific task.

3 The Design Stage

This section is concerned with creating a hardware and software platform where an experienced programmer can program task machinery.

A programmer would use the platform created to generate a user interface to extract task information from an expert in the task being automated. The four levels in the machine design stage, which will provide the platform for creating a task machine, are:

- Joint / MicroComputer Level. This is the lowest level within the hierarchy. The emphasis at this level is on driving the machinery motors to specified positions or velocities and conveying sensor information to higher levels in the system.
- Point-to-Point Level. This is the level of programming that would be achieved using a joystick or similar device. The control at this level might include directional linear and velocity-controlled motion. Within the philosophy adopted here, if aspects of point-to-point control are not required by the task machine then only the required form of motion should be available, for example, circular or continuous path. This terminology is designed to specify the level of control, not the specific application.
- Primitive Motion Level. This linguistic level is a form of point-to-point programming but using textual data entry. The sensor information for the application should be available at this level. This level is a _stepping stone' to providing the versatility of the next level.
- Structured Programming Level. The structured programming level should have the functionality to enable an experienced programmer to program the machinery. The programming environment should provide high level structured programming, data handling, and the ability to quickly generate user interfaces. The requirements for this type of programming environment are summarized in [3].

4 The Installation Stage

This stage is used by both an experienced programmer and someone who is an expert at the task. The programmer develops a user interface to allow the application expert to perform the task. The programmer can then develop rules to perform some of the actions the application expert is performing. The rules that are generated would include providing advice on prospective actions. The rules are then presented back to the application expert, through the user interface, to check their validity. As the rules are defined they can be collected together as sub-tasks and displayed for use in the user interface. Eventually, as higher and higher sub-tasks are defined the programming environment can collapse into just one single task. The programmer needs to be focused on extracting the information from the application expert, and the application expert has to be focused on the task that is being performed so that the task rules can be defined and verified. Within this stage there is only one level:

Fig 3 Changing the direction of the wheelchair to avoid an obstacle[32]

The programming environment created by the programmer and used by the application expert should not detract from the task, but instead should focus the application expert onto the task.

The Sub-Task Level. Sub-tasks are operations that will be used within the final task, for example; changing the direction of the wheelchair to avoid an obstacle (Fig 3), reprogramming the wheelchair controller parameters during operation, could be classed as sub-tasks for an overall operation. This level is concerned with using a structured programming environment to capture and verify rules from an application expert, and then to implement the rules in the form of sub-tasks. These sub-tasks would eventually disappear as the rules determining the use of the sub-tasks developed. The sub-tasks need to be verified with an application expert as they are developed, to check their validity.

5 The Execution Stage

This stage is the highest in the system. After the task rules have been verified, the sub-task level collapses and becomes the task level. The task level forms the only level in the execution section for powered wheelchairs. All the programming of the machinery is performed directly from the information supplied by the user.

Task Level. The task machinery at this level is able to automatically perform specific tasks and also able to give advice on proposed actions. The level would interface to a multi-task level where it would receive coordination information for integration with other machines in a real world environment. An interface would also be provided to enable the machine to advise on the validity of proposed actions. An optional user interface would also be available for monitoring performance and for direct task level programming.

6 Discussion

It has been suggested in the previous section that the programming environment for the creation of a task machine should be designed for an experienced programmer. Any programming environment used for this purpose should therefore assist the programmer by providing the tools and facilities necessary for the task. The main functions the programmer would be required to perform are:

- To control the powered wheelchair.
- React to sensory information (including wheelchair user inputs)
- Communications with other devices in the system.
- Development of a user interface to capture and verify the task rules.

A programming environment that is suited to the task is of greater assistance to the programmer than one that is not. This also applies to the physical structure of the machinery. The complexities of the system will be reflected in the programming environment and any over complication will detract the programmer from the task. Similarly, if the programming environment is under-equipped for the task then this will also detract.

The structured programming environment used to create two virtual task machines is described there along with a graphical resource that was used in the structured programming environment to generate user interfaces.

The use of new user interfaces such as virtual reality systems [7] provide a rich platform and it is important to assess the user interfaces, to discover whether these will focus a user onto a task or detract from the task.

Some guidelines for programming languages are clarity, simplicity and unity of language concept. A programming language should be an aid to the programmer. It should provide them with a clear, simple, and unified set of concepts that can be used as primitives in developing algorithms. To this end it is desirable to have a minimum number of concepts, with the rules for their combination being as simple and as regular as possible. It is this semantic clarity, and clarity of concept, that are most significant in determining the value of a language.

Clarity of program syntax affects the ease with which a program may be written, tested, and later understood and modified. This readability is another important aspect in determining the value of a language.

The language should provide appropriate data structures, operations, control structures, and a natural syntax for the problem to be solved. This extends to the particular task being programmed, and the language should be equipped for the task to be performed. A language particularly suited to a certain class of application will greatly simplify programming.

Abstraction is the ability to get away from the direct commands of the language and to `tailor' the language for the specific task. A language may be abstracted to a slightly higher task specific level through the versatility of data structures and the use of functions and procedures.

The reliability of programs written in a language is essential which is particularly true for wheelchair programs. There are many techniques for verifying that a program correctly performs a task. These range from graphical methods, to running the code and checking the result and structured testing.

The portability of programs is a useful aid for applications that involve similar tasks. The ability for both the programs written and the languages themselves to be transferred from one type of machine to another may have huge benefits.

The approach presented in this paper has built on the fundamental principle that a user should only program using familiar concepts and terminology. Lower level programming should be performed by an experienced programmer, whose focus should be on extracting expert knowledge from the user and reflecting that knowledge back through the user interface for verification.

Future work will investigate wheelchair veer [10,37], web interfaces [38,39], assessment using multi-media [40] Perception [41] and automation [42].

References

1. Strickland, P.: Task oriented robotics, (PhD), Portsmouth Polytechnic, UK. (1992)
2. Tewkesbury, G.: Design using distributed intelligence within advanced production machinery, (PhD), University of Portsmouth, UK. (1994)
3. Tewkesbury, G.: Task programming for powered wheelchairs, In: & Paper, E (ed.) Journal of intelligent mobility, vol. 14, pp. 247-250. (2011)
4. Stott, I., Sanders, D.: A new prototype intelligent mobility system to assist powered wheelchair users. Industrial Robot 26 (6), 466-475 (1999).
5. Goodwin, MJ., Sanders DA., Poland GA.: Navigational assistance for disabled wheelchair-users. Euromicro Conference 95 Volume: 43 73-79 (1997).
6. Stott, I., Sanders, D.: New powered wheelchair systems for the rehabilitation of some severely disabled users. Int' Journal of Rehabilitation Research 23 (3), 149-153 (2000).
7. Stott, I., Sanders, D.: The use of virtual reality to train powered wheelchair users and test new wheelchair systems. Int Journal of Rehab Research 23 (4), 321-326. (2000).
8. Sanders, DA., Bausch, N.: Improving Steering of a Powered Wheelchair Using an Expert System to Interpret Hand Tremor. Proc of Intelligent Wheelchairics and Applications (Icira 2015), Pt Ii, vol. 9245, pp. 460-471 (2015).
9. Sanders, DA.: Using self-reliance factors to decide how to share control between human powered wheelchair drivers and ultrasonic sensors, IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 25, no. 8, pp. 1221-1229 (2017).
10. Sanders, D., Langner, M., Tewkesbury, GE.: Improving wheelchair-driving using a sensor system to control wheelchair-veer and variable-switches as an alternative to digital-switches or joysticks. Industrial Robot 37 (2), 157-167 (2010).
11. Sanders, D., Tewkesbury, GE., Stott, IJ., Robinson, DC.: Simple expert systems to improve an ultrasonic sensor-system for a tele-operated mobile-robot. Sensor Review 31 (3) 246-260 (2011).
12. Sanders, DA., Graham-Jones, J., Gegov, A.: Improving ability of tele-operators to complete progressively more difficult mobile robot paths using simple expert systems and ultrasonic sensors. Industrial Robot-an International Journal 37 (5), pp: 431-440. (2010).
13. Sanders, DA.: Non-Model-Based Control of a Wheeled Vehicle Pulling Two Trailers to Provide Early Powered Mobility and Driving Experiences. IEEE Transactions on Neural Systems and Rehabilitation Engineering 26 (1), 96-104 (2018).
14. Sanders, D., Gegov, A.: Using artificial intelligence to share control of a powered-wheelchair between a wheelchair user and an intelligent sensor system, EPSRC (2018).
15. Sanders, D.: Comparing ability to complete simple tele-operated rescue or maintenance mobile-robot tasks with and without a sensor system. Sensor Review 30(1), 40-50 (2010).
16. Sanders, DA., Langner, M., Gegov, A., Ndzi, D., Sanders HM., Tewkesbury GE.: Tele-operator performance and their perception of system time lags when completing mobile robot tasks, Proc 9th Int Conf on Human Systems Interaction, pp. 236-242 (2016).
17. Sanders, D.: Comparing speed to complete progressively more difficult mobile robot paths between human tele-operators and humans with sensor-systems to assist. Assembly Automation 29 (3), 230-248 (2009).
18. Sanders, DA., Stott, I., Robinson, DC., Ndzi D.: Analysis of successes and failures with a tele-operated mobile robot in various modes of operation. Robotica 30, 973-988 (2012).
19. Sanders, DA., Ndzi,, D., Chester, S., Malik, M.: Adjustment of Tele-Operator Learning When Provided with Different Levels of Sensor Support While Driving Mobile Robots. Proceedings SAI Intelligent Systems Conference 2016, Vol 2 -16, 548-558 (2018).
20. Sanders, DA., Tewkesbury, GE.: A pointer device for TFT display screens that determines position by detecting colours on the display using a colour sensor and an Artificial Neural Network. Displays 30 (2), 84-96 (2009).
21. Sanders, D.: Environmental sensors and networks of sensors. Sensor Review 28 (4), 273-274 (2008).
22. Sanders, D.: Controlling the direction of "walkie" type forklifts and pallet jacks on sloping ground. Assembly Automation 28 (4), 317-324 (2008).
23. Sanders, DA., Baldwin, A.: X-by-wire technology. Total Vehicle Technology 3-12 (2001).
24. Stott, IJ., Sanders, DA., Goodwin, MJ.: A software algorithm for the intelligent mixing of inputs to a tele-operated vehicle. Euromicro Conference 95, Volume: 43, 67-72 (1997).
25. Sanders, D.: Analysis of the effects of time delays on the teleoperation of a mobile robot in various modes of operation. Industrial Robor 36 (6), 570-584 (2009).
26. Pellegrini, N., Guillon, B., Prigent, H., Pellegrin,i M et al.: Optimization of power wheelchair control for patients with severe Duchene muscular dystrophy. Neuromuscular disorders 14 (5), 297-300. (2004).
27. Taylor, PB., Nguyen, HT.: Performance of a head-movement interface for wheelchair control. Proc' 25th Int Conf of IEEE Eng in Medicine and Biology Society, Vols 1-4 and… A new beginning for human health 25, 1590-1593 (Part 1-4) (2003).
28. Gosain, D., Jyoti, D., Asiwal, D., Singh, S., et al.: Design and development of a foot controlled mobility device. Proc' of 2nd Frontiers in Biomedical Devices Conf, pp83-87. ISBN: 978-0-7918-4266-9 (2007).
29. Bergasa-Suso, J, Sanders, DA, Tewkesbury, GE.: Intelligent browser-based systems to assist Internet users. IEEE Transactions on Education 48 (4), 580-585. (2005).
30. Sanders, DA., Bergasa-Suso, J.: Inferring Learning Style From the Way Students Interact With a Computer User Interface and the WWW. IEEE Trans on Ed 53 (4) 613-620 (2010)
31. Sanders, D.: Viewpoint - Force sensing. Industrial Robot 34(4), pg 177. 268 (2007).
32. Langner, M.: Effort Reduction and Collision Avoidance for Powered Wheelchairs; sensor Assistive Mobility System, (PhD), University of Portsmouth, UK. (2012)
33. Chang, YC., Yamamoto, Y.: On-line path planning strategy integrated with collision and dead-lock avoidance schemes for wheeled mobile robot in indoor environments. Industrial robot – an international journal 35 (5), 421-434 (2008).
34. Sanders, DA.: Real time geometric modeling using models in an actuator space and Cartesian space. Journal of Robotic Systems 12 (1) 19-28 (1995).
35. Erwin-Wright, S., Sanders, D., Chen, S.: Engineering Applications of Artificial Intelligence 16 (5-6), 465-472 (2003).
36. Urwin-Wright, S., Sanders, D., Chen, S.: Terrain prediction for an eight-legged robot. Journal of Robotic Systems 19 (2) 91-98 (2002).
37. Langner, MC., Sanders, DA.: Controlling wheelchair direction on slopes. Jrnl of Assistive Technologies. 2 (2), 32-42 (2008).
38. Bergasa-Suso, J, Sanders, DA, Tewkesbury, GE.: Intelligent browser-based systems to assist Internet users. IEEE Transactions on Education 48 (4), 580-585. (2005).
39. Sanders, DA., Bergasa-Suso, J.: Inferring Learning Style From the Way Students Interact With a Computer User Interface and the WWW. IEEE Transactions on Education 53 (4) 613-620 (2010)
40. Chester, S., Tewkesbury, G., Sanders, D.,et al.: New electronic multi-media assessment system. Web Information Systems and Technologies 1, 414-420 (2007).
41. Sanders, DA.: Perception in robotics. Industrial Robot 26 (2) 90-92 (1999).
42. Sanders, DA. Rasol, Z: An automatic system for simple spot welding tasks. Total Vehicle Technology Conference 263-272 (2001).