Assessing the Use of Continuous-Time and Timed-Triggered Models for Designing Cyber-Physical Systems

Fernando Silvano Gonçalves
and Leandro Buss Becker
Department of Automation and Systems (DAS)
Federal University of Santa Catarina (UFSC)
Florianópolis, SC, Brazil.
Email: fernando.goncalves@posgrad.ufsc.br
leandro.becker@ufsc.br

***Abstract*—Different models of computation can be used for designing the embedded computing system from a Cyber-Physical System. Such embedded system should be in charge of executing the control algorithms and communicating with the sensors and actuators. Timed-Trigger and Continuous-Time are examples of models-of-computation (MoC) used in such application domain. This work presents an assessment of using these two MoCs when applied for representing the stability control system from an Unmanned Aerial Vehicle. The paper addresses facilities and difficulties on mapping high-level models using these two MoCs to a typical execution platform.**

I. Introduction

Cyber-Physical Systems (CPS) are characterized for creating a connection between computational devices and the real world. While computational devices are in charge of data processing, the real world refers to the electromechanical processes that can be controlled by the computers [1]. The CPS can be organized into a single centralized application or split into subsystems in order to meet the environment requirements that they are applied [2].

Designing CPSs is a complex task, especially due to the required adjustments in the timing characteristics of the respective software, which must be in compass with the physical process. It is observed that two different models-of-computation (MoCs) are usually applied to fulfill these needs, the Continuous-Time (CT) and Timed-Triggered (TT) models.

The CT model is a composition of actors that represent a set of mathematical functions of continuous nature. Given its continuous nature, the mathematical functions could not be executed in a discrete time computer without modifications. Therefore, the approximate execution of this model are performed by a solver that numerically approximates the solution of an ordinary differential equation (ODE) [3].

The CT model can be seen as a synchronous-reactive system where the time interval between the reactions is determined by the solver. In other words, the underlying infrastructure consults the actors in a determined time interval such that the events of interest can be precisely collected.

Differently, in the TT model presented in [4] there exists a distributed mechanism to trigger events periodically in accordance with a clock signal that measures the elapse of time. This model results in the so-called Time-Triggered Architecture (TTA), which contains a set of model artifacts used to support and coordinate different kinds of embedded real-time systems.

The TT MoC is similar to the synchronous-reactive in the sense that a global clock coordinates the computations. The TT MoC associates every computational processing with a temporal logic execution. The inputs are obtained in ticks of the global clock, however the outputs are only visible to other components in the next tick. In between tick times the components do not communicate among themselves, avoiding problems like race conditions [3].

In the present paper we aim at assessing the use of the CT and the TT MoCs for designing the stability control system of an Unmanned Aerial Vehicle (UAV), which is an example of a typical CPS. The paper also addresses facilities and difficulties on mapping the high-level models representing these two MoCs to a typical execution platform.

The remaining parts of the paper are organized as follows: Section II discusses in more details the MoCs to be applied for designing the UAV stability control. Section III details the UAV stability control used as case study. Section IV presents the TT model designed to support the UAV system. Section V describes the CT model developed from the aircraft. Section VI discusses about the model of computation comparing their characteristics and analyzing their behavior. Finally, Section VII concludes the paper.

II. Models of Computation

Cyber-Physical Systems are typically designed by selecting and connecting a set of components. These structures should be integrated between themselves in order to guarantee the iteration with the physical process. In the real environment process running together and the system components can be able to operate simultaneously. There is no particular order in their operations [3].

The Models-of-Computation (MoCs) provide a framework to support the implementation of CPSs. Different kinds of MoCs can be adopted according to the characteristics of the CPS under construction. Since in this paper we concern with the stability control system of an UAV, it focusses the Time-Triggered (TT) and the Continuous-Timed (CT) models, as further detailed in the incoming subsections.

A. *Timed-Triggered Model*

The TT model defines that computations should be closely related with a periodic clock [5]. To implement the TT model a set of components are needed: the RT Entity (RTE), the Interface, the Communication System (CS), and the Transducer.

The RTE describes the system actor actions, being responsible for the data processing. For instance, this component implements the functions of the system and produces the data to be consumed by other components. An interface is applied to this component in other to allow the exchanging of information with other Entities and Transducers of the model.

The Transducer is responsible for providing the communication, allowing data to be exchanged between the TT system and the environment. In other words, it provides the communication protocols that are needed to support the process of exchange information with the real world. Only by this component the TT model can communicate with the environment where it is applied.

RTEs and Transducers make use of an Interface, which is responsible for the information exchange between the system components. Each interface stores the data produced by the components. It is using Interfaces that data is transmitted between the system actors.

The data exchange between the Interfaces is managed by the CS, where a set of rules need to be specified in order to describe the system information flows. The CS manages the system interfaces and updates the inputs and output of the interfaces according these rules.

B. *Continuous-Time Model*

Systems designed using the CT model are based on ordinary differential equations (ODEs) and can be described as an interconnection of actors, where the communication between these actors occurs by means of signals flowing continuously in time [3].

CT models cannot be implemented without modifications in a digital computer since a digital computer is not capable to operate in continuous time - recall that digital systems are discrete machines. However, the CT model can be modified by using a solver that constructs a numerical approximation of the ODE solution. Different kind of solvers can be used to implement this approximation, such as the Euler Solver.

These approximations are based on a step time, which is used to estimate the function values at some time points. However, this approach presents approximation errors that accumulate over time. Such problem can be overcome by using two different techniques: (i) applying a variable-step solver that defines the step size based on the estimated error, so that a small error is kept; or (ii) implementing a more sophisticated solver that uses the slope of the curve to apply an trapezoidal approximation.

The CT model can be considered as a synchronous-reactive model with a time step between global reactions settled by a solver. This model is represented by a network of actors, where each one is a cascade composition of a simple memory less computation actor and a state machine, and where the actor reactions are simultaneous and instantaneous. The reactions time are determined by a solver [6].

Generally, in every time step the solver consults the actors in order to precise the system events, like level crossings. Despite the additional effort to provide a solver, the mechanisms required to achieve a CT model are not much different from those required to achieve a Synchronous-Reactive and Discrete Event models.

Different software tools can be used to implement and simulate CT based systems. The Simulink tool is for sure one of the most widely used tool for such purposes. An alternative tool MATRIXx from National Instruments, which also supports graphical modeling.

In this work both CT and TT models were developed for representing the UAV control system that is described in the next section, together with the respective models.

III. UAV Description

The UAV used in this study was created in the context of project ProVant[1], which is currently being developed at the Federal University of Santa Catarina (UFSC) in partnership with the Federal University of Minas Gerais (UFMG). The main goal of Provant is to develop an UAV that can be used as a study platform. Currently, different research projects are developed based on this aircraft, such as the development of control strategies for stability and path track, communication with a wireless sensors network, load transportation, use of intelligent agents and others.

The developed UAV has a Vertical Takeoff and Landing (VTOL) Tilt-rotor configuration. Figure 1 show the prototype of the aircraft, which is composed by two rotors that can be rotated longitudinally up to 180 degrees and by the actuation of individual servomotors.

For control purposes three structures are considered on the UAV: the main body (frame B); the right rotor (frame R); and the left rotor (frame L). The longitudinal gradients performed by the servomotors are described by the angles α_R to the right rotor and α_L to the left rotor.

The UAV dynamic model was obtained using Euler-Lagrange equations as described in [7]. It gives a mathematical representation that describes the UAV's motion in relation to the forces and torques applied to it. This model was used to create a simulation environment in Simulink, in which the control algorithms can be tested.

A control strategy for a similar UAV is presented in [8], where the authors proposed a control approach based on three blocks: the translational control that performs the path tracking of the UAV; the rotational control that is responsible to stabilize the aircraft; and the signal transformation that performs a transformation of the output control data in signals of force and angle to send to the system actuators. In the present paper only the rotational controller and the signal transformation are discussed.

[1]http://provant.das.ufsc.br

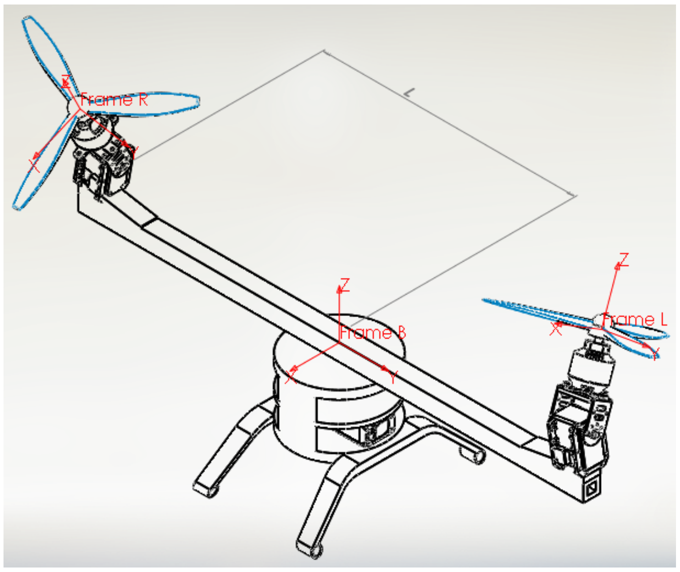


Fig. 1: Tilt-rotor model of UAV.

The rotational controller is responsible for the UAV stability during the flights. The rotational model is described with the following equation:

$$M_1(\Theta)\ddot{\Theta} + C_1(\Theta,\dot{\Theta})\Theta = \tau \tag{1}$$

As the motions are based on the Inertia Matrix (M_1), Coriolis, and the centrifugal force Matrix ($C1$) and the external torque vector (τ). A feedback linearization is applied to this model as follows:

$$\tau = \begin{bmatrix}\tau_\phi\\ \tau_\theta\\ \tau_\psi\end{bmatrix} = M_1(\Theta)\begin{bmatrix}\gamma_1\\ \gamma_2\\ \gamma_3\end{bmatrix} + C_1\left(\Theta,\dot{\Theta}\right)\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} \tag{2}$$

Substituting equation 2 into 1 it results in the new system dynamics $\Gamma = [\gamma_1\gamma_2\gamma_3]$ where γ_i are the PD equations:

$$\begin{aligned}\ddot{\phi} = \gamma_1 &= -K_{p\phi}(\phi-\phi_d) - K_{v\phi}(\dot{\phi}-\dot{\phi}_d) + \ddot{\phi}_d,\\ \ddot{\theta} = \gamma_2 &= -K_{p\theta}(\theta-\theta_d) - K_{v\theta}(\dot{\theta}-\dot{\theta}_d) + \ddot{\theta}_d,\\ \ddot{\psi} = \gamma_3 &= -K_{p\psi}(\psi-\psi_d) - K_{v\psi}(\dot{\psi}-\dot{\psi}_d) + \ddot{\psi}_d.\end{aligned} \tag{3}$$

Where K_p and K_v are the control parameters.

The external force equations are:

$$F_{zb} = f_R\cos(\alpha_R) + f_L\cos(\alpha_L) \tag{4}$$

$$\tau = \begin{bmatrix}\tau_\phi\\ \tau_\theta\\ \tau_\psi\end{bmatrix} = \begin{bmatrix}[f_L\cos(\alpha_L) - f_R\cos(\alpha_R)]\,l\\ [f_L\sin(\alpha_L) + f_R\sin(\alpha_R)]\,h\\ [f_L\sin(\alpha_L) - f_R\sin(\alpha_R)]\,l\end{bmatrix} \tag{5}$$

Where F_{zb} is the force in the Z axis and the control inputs of the system are f_R, f_L, α_R, and α_L.

To apply the controller from equation 2, equation 5 must be solved in relation to the control inputs. This leads to:

$$\begin{aligned}f_L &= \frac{1}{2}\sqrt{\left(\frac{\tau_\theta}{h}+\frac{\tau_\psi}{l}\right)^2 + \left(F_{zb}-\frac{\tau_\phi}{l}\right)^2},\\ f_R &= \frac{1}{2}\sqrt{\left(\frac{\tau_\theta}{h}-\frac{\tau_\psi}{l}\right)^2 + \left(F_{zb}+\frac{\tau_\phi}{l}\right)^2}.\end{aligned} \tag{6}$$

$$\begin{aligned}\alpha_L &= \arctan\left(\frac{\frac{\tau_\theta}{h}+\frac{\tau_\psi}{l}}{F_{zb}-\frac{\tau_\theta}{l}}\right),\\ \alpha_R &= \arctan\left(\frac{\frac{\tau_\theta}{h}-\frac{\tau_\psi}{l}}{F_{zb}+\frac{\tau_\theta}{l}}\right).\end{aligned} \tag{7}$$

Where l is the fuselage length and h is the center of mass displacement in the Y axis. As we only want to stabilize the UAV, F_{zb} will be the force required for the aircraft to maintain is current altitude, which is the product *mass*gravity*. More details can be seen in [8].

In order to support the control strategy the hardware architecture was specified. This structure is composed by two embedded platforms: the Olimex STM32F4 [9] and the Beaglebone [10]. The first is responsible to communicate with the sensors and actuators and the second performs the high level processing, running the control algorithms based on the received data from the first platform.

Coupled to the embedded platforms there is a set of sensor and actuators: one Inertial Measurement Unit (IMU); one Global Position System (GPS); two Servomotors; two Electronic Speed Controller (ESC); one Sonar; one Radio Control; and, finally, the Radio Wireless.

The arrangement of these components within the platforms is presented in Figure 2. At this time only the rotational controller was implemented and just part of the components are used. This approach uses the IMU, servomotors, and ESCs. In the incoming phases of the project the UAV should be capable to perform autonomous flights and, therefore, all components must be used.

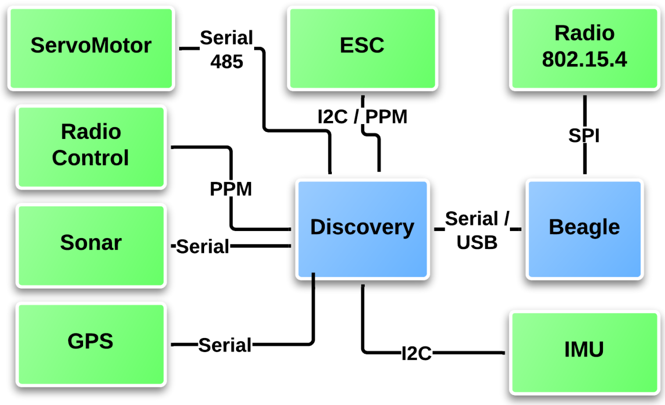


Fig. 2: Hardware Structure of UAV.

Based on the UAV description the two MoCs were developed in order to support the UAV system, running the control algorithms and maintaining the stability of the aircraft during the flights.

IV. Time-Triggered Model

UAVs are critical embedded systems that need periodical message responses in order to guarantee the aircraft stability. To assure these messages a Time-Triggered model was developed. The TT model designed to represent the UAV system is composed of the following components: two Transducers (imuThread e actThread), one RT Entitie (controlThread), one communication system (communicationSystemThread), and three Interfaces (intAct, intIMU, intCE). The class diagram of this model it is depicted in Figure 3a.

The TT model designed to represent the UAV system is composed of the following components: two Transducers (imuThread e actThread), one RT Entitie (controlThread), one communication system (communicationSystemThread), and three Interfaces (intAct, intIMU, intCE). The class diagram of this model it is depicted in Figure 3a.

The *actThread* Transducer is responsible for interfacing the high-level platform (where the control system is executed) with the low-level platform (that interacts with the sensors and actuators). It basically sends the control references to the system actuators (servomotors and ESCs). This component was implemented using a periodic thread with a period of 6 milliseconds.

Its counterpart is the *imuThread* Transducer, which receives the estimated position data from the low-level platform. This information is used in the control algorithms for path-following. This Transducer was also developed using a periodic thread with period of 6 milliseconds.

The *controlThread* RTE was developed to execute the control algorithms responsible to maintain the aircraft stability. This component receives the attitude and angular velocity from the *imuThread* Transducer, applies the control algorithms and sends the control references to the *actThread* Transducer. This entity was also developed in a periodic thread with a period of 6 milliseconds.

The Transducers and the RTE exchange information by the use of interfaces, which are responsible to create a communication channel between the system components. These components are implemented in global objects that are managed by the CS.

The information flow was specified in the CS, at this component all the system connections are mapped by the definition of rules that connect the inputs and outputs of the system interfaces. By this rules different outputs from different interfaces can be connected with the same interface, as well as the output of one interface can be sent to different interfaces simultaneously.

The CS is also responsible for ensuring the delivery of the messages exchanged between the interfaces. The critical sections are protected by the using mutexes, barriers, and shared variables. The information exchanged is periodic and this component was modeled in a periodic thread with a period of 6 milliseconds. By the use of shared variables, each loop the new produced data are signalized and exchanged by a set of predefined rules.

The TT system is controlled by a global clock, where in each loop new data is received from the IMU. Such data is sent to the RTE by the CS and after that the control references are calculated. With a new produced data in the RTE, the CS sends this information to the transducer responsible for the actuation and the calculated control references that are sent to the second platform in order to act in the UAV system.

The TT architecture was implemented using the C++ language and the *Pthreads* library. It runs in the Beaglebone platform, which uses the Angstrom Linux [11].

V. Continuous-Time Model of UAV

The CT Model of the UAV was developed based in a model implemented in the Simulink tool for simulation purposes. Such initial model allowed to analyze the system behavior and the model structure that will be applied to the embedded platform. This system was implemented in the Beaglebone and interfaced with the low-level platform in order to guarantee the estimation of the attitude and angular velocity, plus the additional actuation commands.

The CT model structure is composed by a set of threads, as follows: the control thread, the communication thread, and the main thread. The system architecture is represented by the class diagram illustrated in Figure 3b.

The control thread is responsible for implementing the stabilization control, as described in equations 2-5. Coupled with the rotational motion equation a signal transformation was applied in other to obtain the references by system actuators, this transformation is described by equations 6 and 7. This object was designed in a periodic thread with a period of the 6 milliseconds.

The communication thread was created in order to establish a communication channel with the system sensors and actuators. This component receives the estimation of the attitude and angular velocities of the aircraft from the low-level platform and sends the control references to the system actuators. These information are sent to the control thread by the use a global shared object that can be accessed by both tasks. This thread was implemented in a periodic thread with a period of 6 milliseconds.

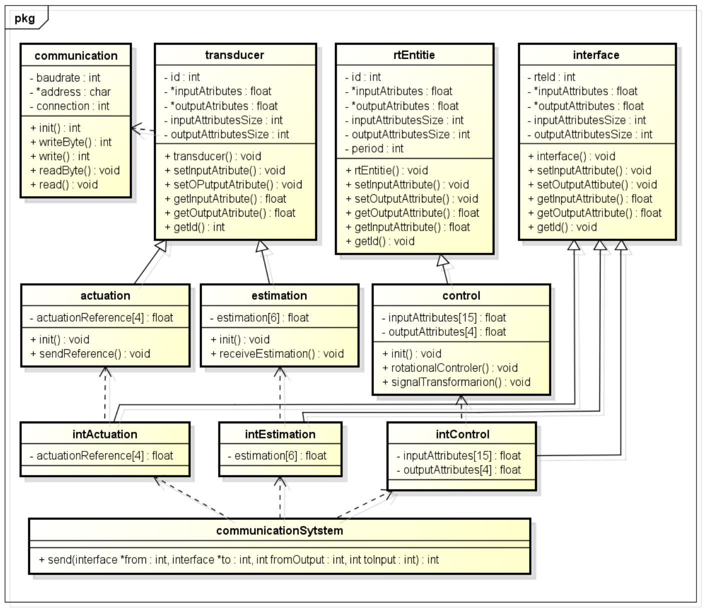
Due to the fact that the real-time operation system (RTOS) was not used, in this approach the main thread was implemented in order to schedule the other tasks according to their periods, ensuring the system periodicity. The threads are managed by the use of semaphores, and the main thread signalizes the periodic execution of the tasks.

As previously described, the data exchanged by the system components was represented by the use of global objects that maps the system inputs and outputs. These components are protected by the use of a mutex.

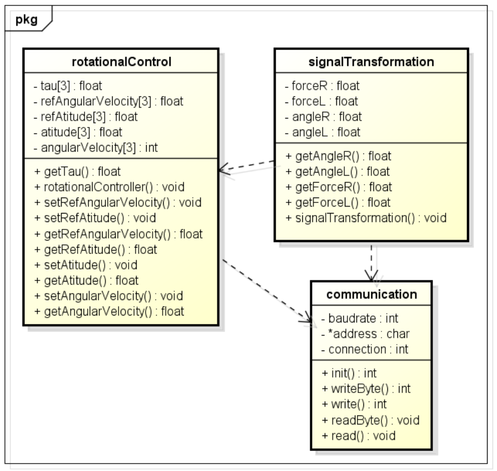
Based on the specifications the system, it was implemented and tested in order to verify the behavior of both systems and the UAV. Follows a comparative analysis of these approaches describing their characteristics.

VI. Comparative Analyzes

In order to perform a comparison between these two modeling approaches (TT and CT), several evaluation criteria have been identified. These criteria are specified in order to



(a) Timed-Trigger Model of UAV.



(b) Continuous-Time Model of UAV.

Fig. 3: Class Diagrams of Models of Computation.

TABLE I: Evaluation Criteria

Criterion	Description	Evaluation	Expressed by
1 - Maintainability	Easiness to make modifications in the models e.g., addition of new elements and changes in the external elements such as sensors.	Qualitative	$**$ strong $*$ adequate 0 weak
2 - Modularity / Hierarchy	Capability of split a large system into independent modules.	Qualitative	$**$ strong $*$ adequate 0 weak
3 - Expressiveness	Capability of the modeling language primitives to describe the specification.	Quantitative	Number of lines of code programmed by the designer.
4 - Simulation	Capability of verifying if the specification can be used to validate the implementation.	Qualitative	$**$ strong $*$ adequate 0 weak
5 - Verification	Capability of demonstrating formally that the specification or generated program meets the requirements.	Qualitative	$**$ strong $*$ adequate 0 weak
6 - Implementability	Criterion for evaluating whether the specification can be easily refined or translated into an implementation, which is compatible with the rest of the system.	Qualitative	$**$ strong $*$ adequate 0 weak

perform a qualitative and quantitative analysis. The set of criteria can be seen in Table I.

The obtained results are summarized in Table II. For evaluating the qualitative aspects, the $**$ symbol was used to indicate a particular strength of the approach, $*$ indicates that the model meets the criterion in a way that is adequate, but less than ideal, and 0 was used to indicate a weakness of the model.

TABLE II: Comparison Results

Evaluation Criteria	TT Model	CT Model
1- Maintainability	$**$	$*$
2- Modularity / Hierarchy	$**$	$*$
3- Expressiveness	1250	1184
4- Simulation	$*$	$**$
5- Verification	$*$	$*$
6- Implementability	$*$	$*$

The evaluation begins by analyzing the maintainability, the intrinsic object orientation (OO) property from TT model, provide by its artifacts, such as the specialization/generalization presented in the entities, interfaces, and transducers. The conclusion is that it provides better maintainability if compared to the CT approach where the model artifacts presented are not formally defined.

Considering modularity/hierarchy aspects, it is possible to conclude that the TT model present a better decomposition than the CT model. This can be observed by comparing the class diagrams of both approaches. The first (Figure 3a) present a set of class that allows an easier interpretation of the physical behavior. Although the CT approach (Figure 3b) presents less classes than the TT approach, the system is also represented in both approaches. However, the use of the less classes does not allow a higher modularity of the system.

The next criteria concerns the model expressiveness, analyzing the number of lines code programmed by the designer in each model, it can be observed that in the TT model the designer has to manually write 1250 code lines, whilst in CT model the designer has to manually write 1184 code lines. The large number of code lines manually written in both model is due to the fact that the code generated in the tools, like Simulink for example, is not suitable to directly application in the embedded platform. For such reason the applications were implemented manually. Analyzing the structures of the models of computations, CT presents less artifacts than the TT model and, consequently, it has less code lines in the model.

Regarding model simulation, it is possible to observe that in order to provide such features suitable modeling/design tools are required. Regarding the UAV system, only the CT model could be simulated by the use of the Simulink. The available tools of TT model simulation could not be located. However, considering the authors experience, the TT model simulation could be performed using a Hardware-In-the-Loop (HIL) environment, allowing to analyze the model behavior before its application in the real environment.

In respect to the capacity to perform any kind of formal verification, by using the assert library of the C++ language, both models provide the support for formal verifications. However, a huge effort should be made by developers in order to specify the system properties to be verified and to identify the points of the code that this assert should be included in order to guarantee the correctly verification.

Finally, considering the model's implementability, one can see that from both models an architecture-independent specification can be derived. This criterion can be analyzed using three different aspects.

The first is the capability of mapping the systems using the models, the TT model present artifacts that facilitate the system mapping by their use, the CT model does not provide artifacts and its system implementation is more straightforward.

Another aspect is the fact that both TT and CT models need a set of software requirements in order to support the systems implementations, such as multi-thread support, critical sections protection (e.g. mutexes, barriers, and condition variables), and operating system support.

The last one refers to the special care with the critical sections of the systems. In the TT model using the interface artifacts this structures provides an implicit protection to the data exchange, while in the CT model a special care should be taken by the developer in order to guarantee the data safety.

VII. Conclusion

This work has presented a comparison between Time-Triggered (TT) and Continuous-Timed (CT) models in order to map the Cyber-Physical Systems to the embedded platforms by their uses. Considering the obtained results, it can be seen that both models present the same performance for requirement specification. An advantage of using the TT model describes that this model can be easily maintained and modulated than CT models.

Due to the fact that the automatic code generated needs several modifications to apply in the embedded platform the model was manually implemented and the expressiveness was verified according the number of code lines written their implementation. Hence, analyzing the code generated was verified that the TT model presents more artifacts than CT model and consequently more code lines. However, due to the fact the both systems using the same libraries of the C++ language, the code line difference between the systems is not so expressive.

By the use of simulation tools like Matlab/Simulink the CT model has an advantage over simulation systems of the TT model. Analyzing the verification aspects both approaches allow to perform this process but as described before it was necessary a large experience of the engineers in other to specify and implement this process in the code application.

Although the CPS system implementability is possible in both models of computation, the map of the planned system to the real embedded application is not a trivial task. The systems present a set of characteristics that are not directly mapped to the both models, however the TT model has a set of artifacts that can facilitating this process.

Analyzing the models of computation and comparing their characteristics can be concluded that both models meet the requirements to support the UAV system. However, the TT model describes a better behavior than CT model in order to properly support Unmanned Aerial Vehicles systems.

References

[1] E. Lee, "Cyber physical systems: Design challenges," in *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, may 2008, pp. 363 –369.

[2] M. Kim, M.-O. Stehr, and C. Talcott, "A distributed logic for networked cyber-physical systems," in *ELSEVIER Journal of Science of Computer Programming (SCP)*, 2012.

[3] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*, 1st ed. Lee and Seshia, 2010. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/794.html

[4] H. Kopetz and G. Grünsteidl, "Ttp-a protocol for fault-tolerant real-time systems," *Computer*, vol. 27, no. 1, pp. 14–23, Jan. 1994. [Online]. Available: http://dx.doi.org/10.1109/2.248873

[5] H. Kopetz, "The time-triggered model of computation," in *Real-Time Systems Symposium, 1998. Proceedings., The 19th IEEE*, Dec 1998, pp. 168–177.

[6] E. A. Lee and H. Zheng, "Leveraging synchronous language principles for heterogeneous modeling and design of embedded systems." in *EMSOFT*, C. M. Kirsch and R. Wilhelm, Eds. ACM, 2007, pp. 114–123.

[7] R. Donadel, G. Raffo, and L. Becker, "Modeling and control of a tiltrotor uav for path tracking," in *19th World Congress of the International Federation of Automatic Control (IFAC 2014)*, 2014.

[8] A. Bhanja Chowdhury, A. Kulhare, and G. Raina, "Back-stepping control strategy for stabilization of a tilt-rotor uav," in *Control and Decision Conference (CCDC), 2012 24th Chinese*, 2012, pp. 3475–3480.

[9] Olimex. (2014) Stm32-h407 specifications. https://www.olimex.com/Products/ARM/ST/STM32-H407/.

[10] BeagleBoard. (2014) Beaglebone website. http://beagleboard.org/Products/BeagleBone.

[11] OpenEmbedded, OpenZaurus, and OpenSimpad. (2014) Angstrom linux. http://www.angstrom-distribution.org/.