Project no. **222422**

Project acronym: **ECCell**

Project title: Electronic Chemical Cell

Instrument: STREP/FET OPEN

Thematic Priority: Theme 3 Information and Communication Technologies

# **Deliverable n. 5.1:**
# **UML specification of design space for ECCells.**

Due date of deliverable: 15. 10. 2009

Actual submission date: 15. 10. 2009

Start date of project: **1.09.2008** Duration: **3 years**

Organisation name of lead contractor for this deliverable:

University of Southern Denmark (SDU), Steen Rasmussen

## 1. Deliverable n. 5.1: UML specification of design space for ECCells.

Connecting the electronic chemical cell with the standardized modeling languages of computer science requires us to define an interface between both worlds. Several patents exemplify the need for a real-time modeling of micro-fluidic structures[1,2,3] to cope with the multiphase properties. Experience has taught us that this interface-specification can become very complex. Thus a declaration was sought that conveys the most important connection-points while allowing detailed physical and chemical modeling to be relegated to plug-in "black box" modules. Any attempt to lift the physics and chemistry directly to the model description level would elevate the task to defining the majority of aqueous solution and surface physics and chemistry in UML, that is clearly at best achievable by the entire world community. The general structure we adopt is therefore to confine the complicated physical and chemical boundary conditions into black boxes with very general names and to exchange these black boxes with new versions, if physics or chemistry are changing.

The vehicle chosen to specify this interface is the Unified Modeling Language (UML) plus a special extension, call UML-real-time (UML-RT) to reflect the fact that dynamic physical and chemical processes are to be described. Though UML is thought to be as comprehensive as possible, the most recent specification[4] is nearly 800 pages long, it still is concentrated on the aspects of software-development. This has the consequences that the available tools largely support the specification of software-routines and less so physical, chemical or other mathematical model structures.

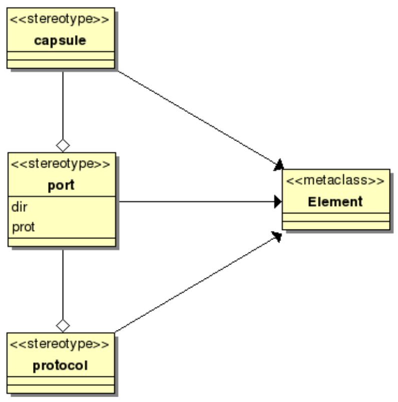
### *1.1 Introduction*

UML is a graphical, object-oriented modelling language that allows one to specify and document the artefacts of software systems. UML is widely used to express the general-purpose software design models and is the *de-facto* standard object-oriented modelling language. One important feature of UML, the raison d'être for choosing this language here, stems from its built-in extensibility mechanisms: stereotypes, tag values and profiles[5]. These allow one to adapt UML to fit the specificities of particular domains or to support a specific analysis. Therefore, numerous UML profiles have been proposed in academia, by industry and/or by (typically government) organizations dealing with standards in order to accommodate the features of real-time software. Real-time software presents, moreover, some specific characteristics. In addition to the timing constraints, an important characteristic of real-time software stems from the interaction with the environment. This interaction is inherently nondeterministic and highly concurrent. UML 2.0 presents some features that support real-time aspects[6]. For example, it supports the modelling of concurrency by providing some concepts, including active objects, concurrent composite states and concurrent operations.

### *1.2 Introduction*

UML-RT is a real-time profile developed by Rational Software7. It uses the UML built-in extensibility mechanisms to capture the concepts defined in the Realtime Object Oriented Modeling (ROOM) Language8. In contrast with the previous profiles, UML/SPT4 and UML/QoS9, UML-RT is not just meant to annotate a design model with information allowing for quantitative analysis. It is a modeling language of its own. Indeed, UML-RT allows the designer to produce models of complex, event-driven and possibly distributed real-time systems. However, UML-RT does not support time and timing constraints modeling. UML-RT is supported by a CASE tool called RationalRT10 that allows for automatic code generation by compiling the models and linking them with a run-time system. UML-RT includes constructs to model the structure and the behavior of real-time systems:

- Structure Modeling: UML-RT provides the designer with entities called capsules, which are communicating active objects. The capsules interact by sending and receiving messages through interfaces called ports. Furthermore, a capsule may have an internal structure composed of other communicating capsules and so on. This hierarchical decomposition allows the modeling of complex systems. The following figure was created with BoUML, a public-domain CASE-tool, see section on BOUML.
- Behavior Modeling: The behavior is modeled by an extended finite state machine and is visualized using UML state diagrams. These state nachines are hierarchical since a state could be lecomposed into other finite state machines. A nessage reception triggers a transition in the state nachine. Actions may be associated with transitions or the entry and/or the exit of a state. Similarly to the wo previous UML profiles, UML-RT lacks formal oundations. UML-RT is, however, a basis for a very ictive research work on schedulability analysis ipplied to real-time software design models.



## 1.3 State of the art

Many publications concerning UML-RT try to formalize this approach [11.] One approach is to map the semantics of UML-RT in to a formal language system [12] to realize a model-driven development for concurrent and distributed applications [13]. A direct specification of a formal language based on UML-RT [14] restricts itself to the formalization of the capsule using the Structure Operational Semantics approach [15.] The question on how to partition active objects on different parallel threads is another issue to be investigated [16,17.] An attempt to consistently describe real-time aspects and resource-restrictions in UML-RT requires a restriction of every communication to the exchange of messages [18]. In addition, actions are specific messages and not operators. Then they require that the messages used in the specified sequence diagrams must be subsets of the messages used in the state-charts and that the mere sequences of these messages must be recognized by the state-machines. This ensures that sequence-diagrams and state-charts can be proven to be syntactical and semantically consistent. Assuring port-compatibility between different capsules [19] is realized using a Failures-Divergence Refinement tool [20,21]. An extension of sequence-diagrams in UML-RT describes broadcasting scenarios [22]. Instead of using UML-RT the specification of the OMG UML-SPT 6 is utilized to describe safety-critical real-time systems[23].

## 1.4 Tool-support

There are many tools available to work with UML-diagrams. The general idea behind these tools is to utilize UML as the only entry-point of software-fragments. A problem recurring very often with software development is the divergence between modeling and accompanying coding of the software. Especially with large and long-lasting projects, the risk of loosing contact between the modeling and the coding is enormous. The reasons are communication problems -- semantic glitches -- and losing people or know-how.

There are essentially two types of tools employed (also termed CASE-tools): commercial tools, like RationalRT (www.ibm.com) and public-domain tools, like BoUML, Umbrello, OMNet++, Eclipse. The advantages of commercial tools are the better support plus powerful features, the advantage of public-domain tools is the long term availability. Especially with projects being developed over many years, availability is the most important issue. Although, in principle, UML-projects can be transported between different tools via the XML Metadata Interchange[24] (XMI) language, the details and versions always differ and, for large projects to be transferred from one tool to the next, they may require complete rewriting. Another advantage of public-domain tools is the independence from any specific company-strategies trying to maximize the companies' profits[1]. For these reasons, we aim at using public-domain base tools and live with the comparatively minor support level supplied by the community.

The following public-domain tools, among many others, are each available for a large number of operating systems:

BoUML[25]: This tool was chosen here because it supports the definition of new profiles and has a stable user-interface. It is supported by SourceForge (http://sourceforge.net/projects/bouml) and still under active development.

Umbrello: This tool is integrated in Linux KDE. Unfortunately, it merely supports UML-1.4 and not UML-2. Profiles cannot be created. The code- and documentation-generation capability seems to be poor.

OMNet++[26]: This tool has a completely different background because it is targeted at modeling networks. Due to the similarity of concurrency and timing constraints between the Electronic Chemical Cell and the network metaphor and because an interface to UML-RT was developed [27] it also seems to be a viable alternative. It was not chosen here because the status of this interface is not really clear.

Eclipse[28]: This is a really large integrated development environment (IDE) and is considered to be the standard for Java-software developers. It also has interfaces to other programming languages and UML-2. Because of being so complex and huge it was not considered at this moment but should be for future larger projects in the Electronic Chemical Cell-domain.

## *1.5 Basic functionalities of electronic chemical cell machines*

The essence of the Electronic Chemical Cell machine is the intimate connection between the local biochemical, biophysical processes, happening concurrently inside a micro-fluidics environment at a particular cell location, and the individually dedicated electronic control hardware on the other side. This strong connection requires a very broad interface with many properties exchanged between the two worlds. A complete modeling of the whole process becomes extremely challenging because so many different effects are playing a role: physics, chemistry, surface-properties, electrical-properties, electro-chemical properties, multi-phase-boarders and information transport processes. A comprehensive language would require one to describe more or less everything that is known in physics, chemistry, electronics, computer-science and biology. This is not possible. Even the restriction to the following language terms would pose a major international undertaking:

### *Information processing properties*

Time, Timeout, Delay, State, Action

[1] Due to diminishing or overtaken companies or licensing fees we have lost several important code-bases in the last 15 years.

### *Physical properties*

Infrastructure: Electrode, Wall

Particles: Molecule, Vesicles, Beads, Ensembles

Geometry: Area, Channel, Volume, Center of ensemble, Position

Mechanical properties: Velocity, Mass, Viscosity, Friction, Force

Thermodynamic properties: Temperature, Pressure

Electrical properties: Charge, Field, Potential, Conductivity

Surface properties: zeta-Potential, Contact angle, Adhesion

### *Chemical properties*

Physico-chemical: pH, Ionic strength, Salt-concentration, Material concentration

Kinetic properties: Forward reaction, Reversible reaction, Catalyzed reaction, Michaelis-Menten kinetic

### *Basic microfluidic functionalities*

Ensemble operations: Fluid focusing, Pulse forming, Mixing, Splitting, Separation

Sequence operations: Interleaving (used with droplets or cells), Sorting (used with droplets or cells), Dosing, Marking

Basic reaction mechanisms for informational molecules:

Informational sequence handling: Cutting, Ligation, Cleavage

Sequence specific interactions: Hybridization, Association, Dissociation, Dissolving, Precipitation

Higher order functional or structural properties: Catalysis through structure, Interference, Gelation, Clustering, DNA-networks

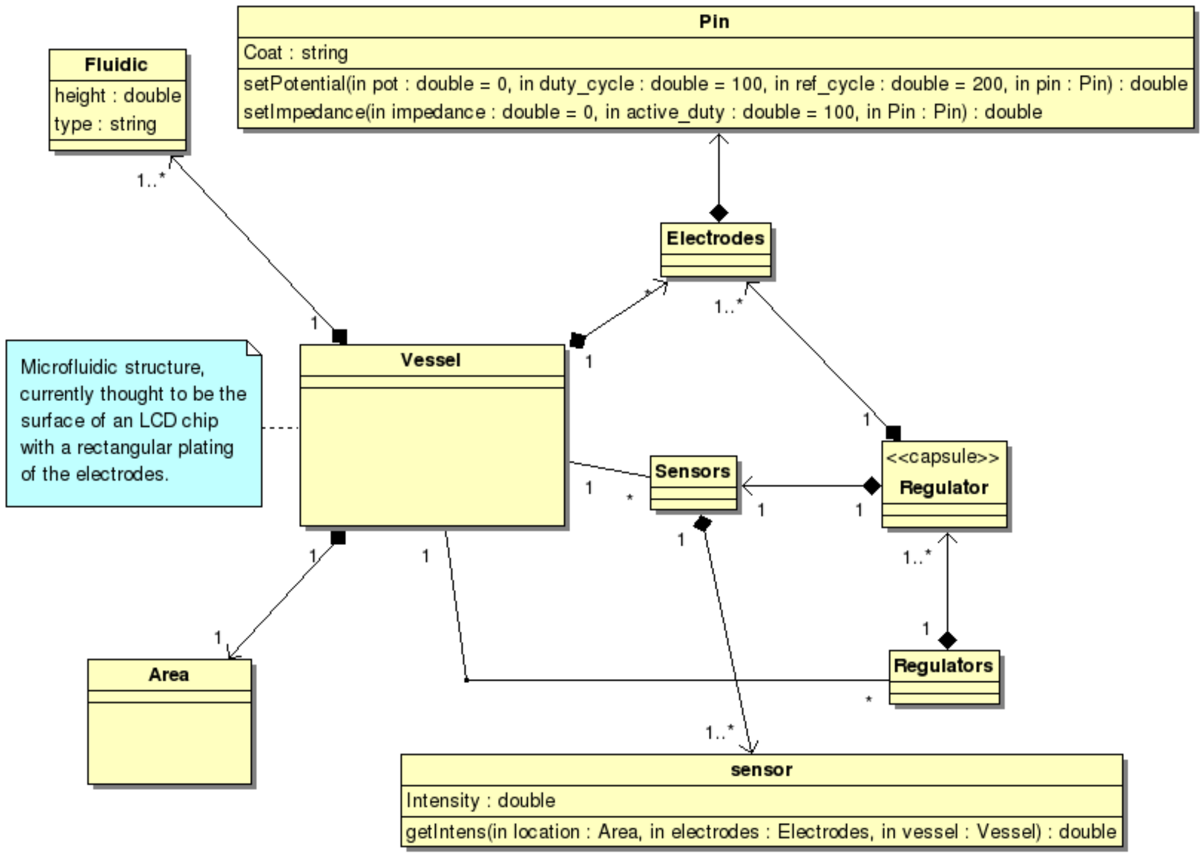
Sequence information management: Coding, Crossover

Additional house-keeping functions: Washing, Shielding, Compartmentalizing, Detection

The tacit assumption behind these terms is not to try a physical, chemical correct mapping of reality into the model but to restrict the modeling to the most important features and boundary conditions. These means that the ab initio predictive capability of such a model would be rather weak but with these terms appropriately parametrized the model should be feasible for usual experiments. As is obvious, in the current state of the project, with only these few man months available for doing the modeling even such a restricted modeling is not possible. Therefore, it was decided to only model the regulatory part in the first instance and to reduce the interface to only electrode-potentials, intensities and impedances. All physico-chemical effects will be subsumed below the operators for getting or setting these properties. Furthermore, it was initially assumed that we do not have a complex micro-fluidic flow-geometry but a simple two-dimensional regular array of rectangular electrodes. This simplification also eases the development of simulation tools that can be utilized for the development of control-algorithms.

## *1.6 The UML-specification realized and the model*

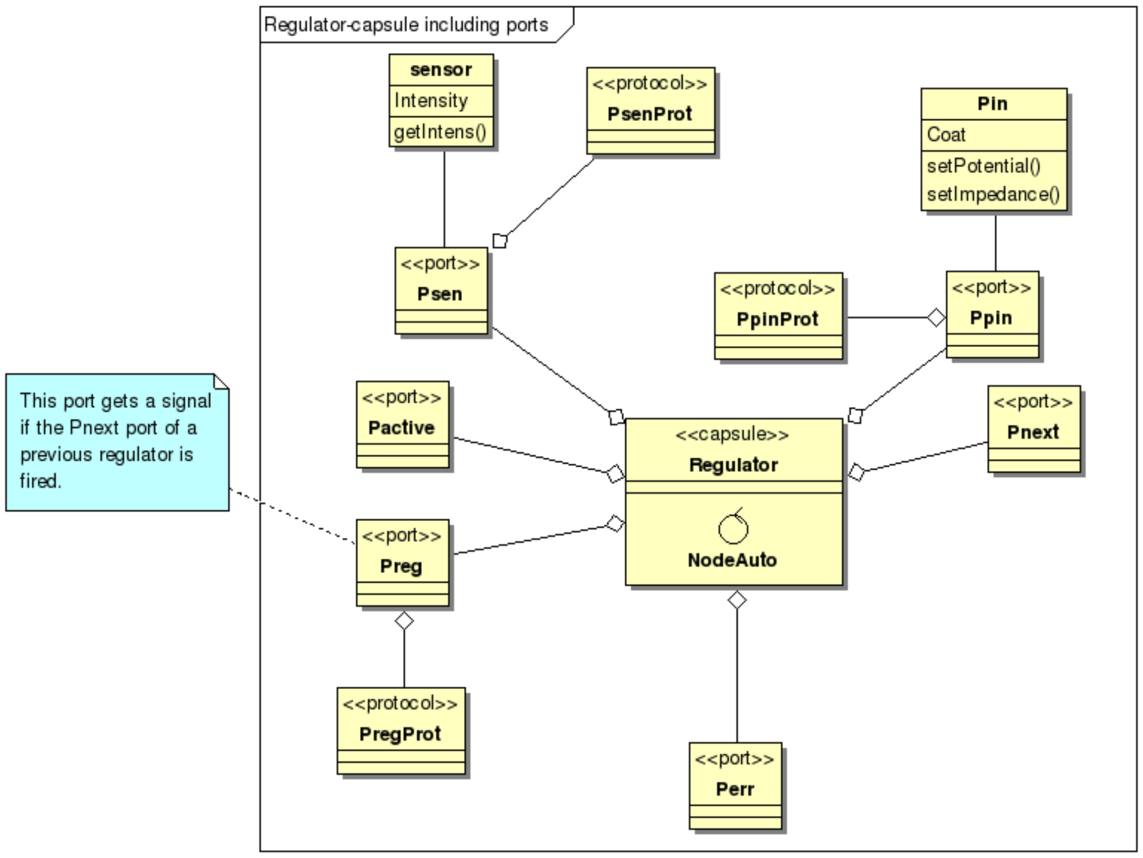
In the following, a sketch of the UML-model will be presented and motivated. The full, so far specified, model is attached as a pdf-file that has been directly generated by the utilized modeling tool BoUML. Even this strongly reduced model is not yet fully functional. Ideally, per bush-button, a fully executable C++ program should be generated which would be able to operate the micro-fluidic hardware. This control-software already exists and to write it a second time would be an intolerable drain on resources. Nevertheless, it is useful to build a UML description of the model, because it clarifies the concepts and points to weaknesses in the current implementation of the control-software.


Pin
Coat : string
setPotential(in pot : double = 0, in duty_cycle : double = 100, in ref_cycle : double = 200, in pin : Pin) : double
setImpedance(in impedance : double = 0, in active_duty : double = 100, in Pin : Pin) : double
Fluidic
height : double
type : string
Electrodes
Microfluidic structure, currently thought to be the surface of an LCD chip with a rectangular plating of the electrodes.
Vessel
Sensors
<<capsule>>
Regulator
Area
Regulators
sensor
Intensity : double
getIntens(in location : Area, in electrodes : Electrodes, in vessel : Vessel) : double

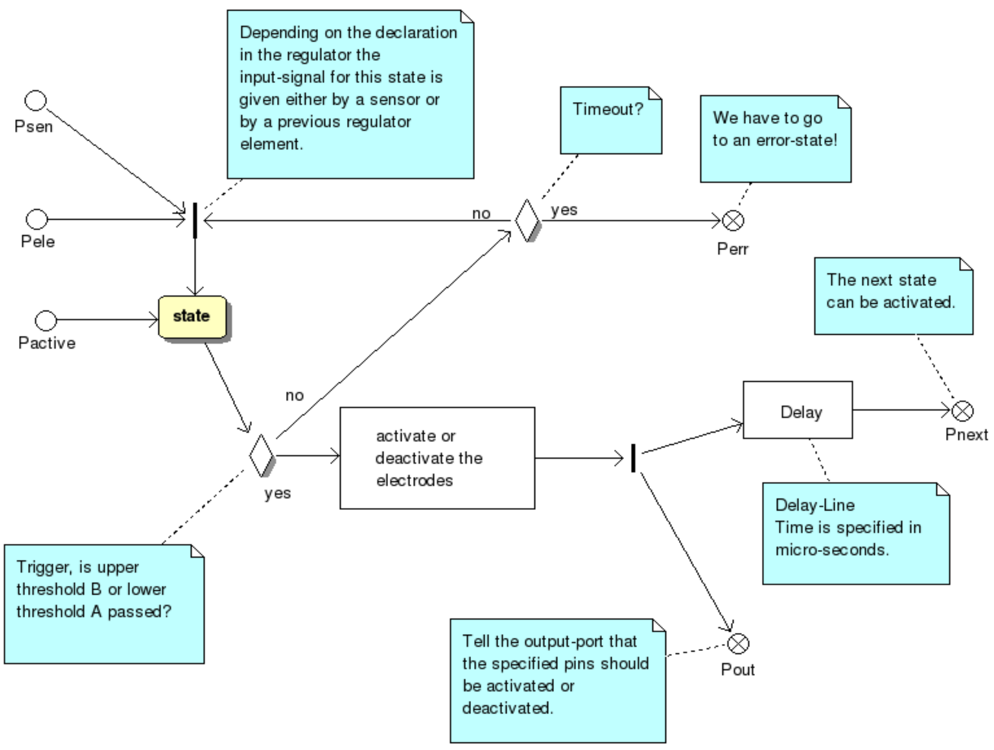
The top-level description of the model is shown in the upper figure. The micro-fluidic system is abstracted via a class called *Vessel*. This vessel is equipped with electrodes, as already mentioned a regular array of rectangular electrodes, an arbitrary number of sensors, additional regulators or controllers and the chemical environment represented by the class *Fluidic*. What happens in the chemistry, or what effects the asserted potentials and impedances will have on physics and chemistry, is not modeled at all. All dynamical chemical and physical changes are subsumed in the black boxes of the intensities of the sensors. This means for example that the operator **getIntens** must somehow reflect internally what happens in the physico-chemical system. In the current implementation, the sensors are derived from fluorescence observed via a special EMCCD camera. A simulator for this system has to accept the following parameters: a set of electrodes, a set of sensors and additional parameters of the system to be simulated. The geometric information is already included in definition of the electrodes and sensors, as can be seen in the figure and the full specification.

To clarify the role of a regulator, the following figure is presented. Per Vessel, arbitrarily many regulators can be defined and operated in parallel. Each capsule is accompanied with ports mediating between the inner world of the capsule and the outside environment.

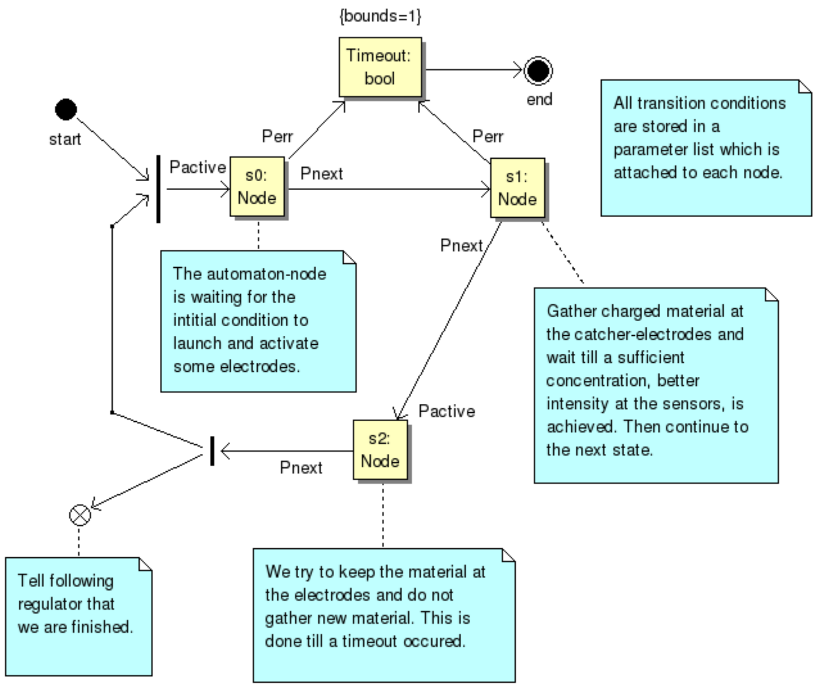
Furthermore, these ports realize a signal transduction. Attached to each port is a protocol. It is assumed here that the ports are only allowed binary communications. The protocol defines which types of signals are allowed.



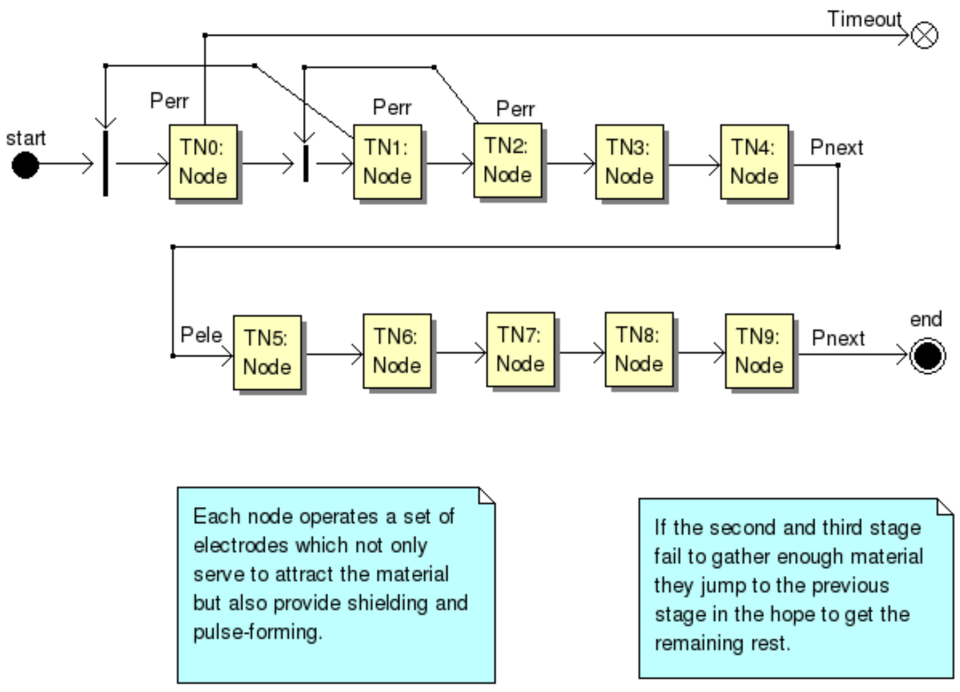
A regulator is a state-machine that looks for the fluorescence intensities to surpass certain thresholds and, in case of an expected event, switches to the next state. Upon switching to the next state, a pre-specified set of pins is activated or deactivated. In the following figure, such an automaton is shown. Its task is to concentrate charged material at a certain region in the micro-fluidic-system via the activation of certain electrodes. When a sufficient intensity is detected, a following regulator is activated to continue with the processing.


Depending on the declaration in the regulator the input-signal for this state is given either by a sensor or by a previous regulator element.
Timeout?
We have to go to an error-state!
Psen
Pele
no
yes
Perr
Pactive
state
The next state can be activated.
no
activate or deactivate the electrodes
Delay
Pnext
yes
Delay-Line Time is specified in micro-seconds.
Trigger, is upper threshold B or lower threshold A passed?
Tell the output-port that the specified pins should be activated or deactivated.
Pout

Each state of this automaton is a state-machine that waits for the arrival of a threshold-crossing-event and then activates some electrodes and provides the signal for the next state of the regulator-automaton. In the following figure, the state's inner state-machine is depicted. As long as no threshold (the upper threshold B or the lower threshold A) is crossed, the inner state-machine cycles in the entry-section. If no event occurs a timeout is generated. When the desired event occurs, a delay-line with a certain delay is activated. After this time, the electrodes are activated or deactivated according to the parameters of the regulator-automaton.


{bounds=1}
Timeout: bool
end
All transition conditions are stored in a parameter list which is attached to each node.
start
Perr
Perr
Pactive
s0: Node
Pnext
s1: Node
Pnext
The automaton-node is waiting for the initial condition to launch and activate some electrodes.
Gather charged material at the catcher-electrodes and wait till a sufficient concentration, better intensity at the sensors, is achieved. Then continue to the next state.
Pactive
s2: Node
Pnext
Tell following regulator that we are finished.
We try to keep the material at the electrodes and do not gather new material. This is done till a timeout occured.

A further example is shown in the following figure. A travelling wave is generated. This is not realized via simply switching electrodes on and off but via smart regulators that detect whether the expected material really arrived. These smart regulators not only detect a malfunction of the travelling wave process, they also are able to realize pulse-forming and thus avoid the inherent problem of dispersion.


Timeout
start
Perr
Perr
Perr
TN0: Node
TN1: Node
TN2: Node
TN3: Node
TN4: Node
Pnext
Pele
TN5: Node
TN6: Node
TN7: Node
TN8: Node
TN9: Node
Pnext
end
Each node operates a set of electrodes which not only serve to attract the material but also provide shielding and pulse-forming.
If the second and third stage fail to gather enough material they jump to the previous stage in the hope to get the remaining rest.

## *1.7 The Electronic Cell and Electronic Genomes*

For us to be able to consider the electronic control structure as “belonging” to the electronic cell, it is necessary to have a definition of cell ownership. The simplest structure of this type is a space associated one: co-localization of chemicals with the region of application of electronic control (electrodes and sensors) control creates the connection between the chemicals and the control program. While other exotic modes of connection (e.g. pattern based) are conceivable, for the moment we intend to stay with this most obvious choice of topo-association. In principle this can be electronically or chemically determined or both. The chemical determination requires a mechanism of deducing a spatial location from a spatial sensor pattern. If there is only one cell in the system, then a sufficient algorithm would find the center of intensity from the sensor array (when fluorescence intensity is proportional to concentration) and then establish a region (e.g. linear, circular or rectangular depending on microfluidic structures) of a fixed size about this center. A more elaborate algorithm would also compute the range of the cell region. An electronic determination may associate the boundary of a cell with a particular pattern of electrode states (a field barrier preventing boundary crossing of key components analogously to a lipid membrane preventing material exchange). When more than one association between chemicals and electronic regulators must be established, this is relatively straightforward in the cases where cell sizes are fixed or the boundaries are clearly recognizable. It is clear that the definition of the boundary codetermines the behaviour of the system, so that simple physical principles are likely to prove the most apt for later evolution.

The model hierarchy above already contains (through inheritance) a location in each of its elements. A topo-association as described above comes into force when for example these locations are treated as relative to a center of mass of replicating molecules. In a particularly simple case, imagine a molecular amplification process combined with an electrophoretic separation, product isolation and split control program. The control program will walk such a set of amplified molecules via travelling wave electrophoresis along a certain path, with its sensors and electrodes virtually staying within a fixed distance of the center of mass of the set of molecules during movement, and the pulse-forming regulators virtually moving together with the majority of molecules. Of course, this process can be codetermined by the chemical properties (migration and reaction rates determining location and quantities) so that for example the initiation time, transport rates and splitting events are codetermined by the chemistry (e.g. splitting via reaction –diffusion non-linearity's in an electric field). Thus these processes can be achieved with a suitable control of electrodes or happen autonomously as a side effect of the regulator algorithms.

Having the moving controllers with the center of masses of molecule collections it is easy to consider also the parameters of these controllers moving together with the molecules virtually. If now, after each separation- or splitting-event these parameters are slightly changed by random perturbations, an evolving population is already instantiated and evolution sets in to optimize the parameters. Parameter sets that are bad for replication yield diminishing sets of molecules, which of course causes the deletion of the moving controllers and their associated parameters. This reflects death in the evolving system. The electronic genome concept thus involves a second type of genetic information associated with the cell: i.e. that involved in specifying the parameters and/or structure of the regulatory control program.

---

1 CA 2582470 A1 (WO 2006/040554 A1) Compartmentalised combinatorial chemistry by microfluidic control Medical Research Council GB, Harvard College US, Priority: 12.10.2004

2 US 2007/0003442 A1, Electronic control of fluidic species Harvard University, Cambridge, Priority: 23.02.2006

3 WO 2007/090531 A1, Arrangement for generating liquid flows and/or particle flows, method for producing and operating said arrangement and use of the latter, Forschungszentrum Karlsruhe GmbH, Priority: 03.02.2006

4 Grady Booch, Ivar Jacobson, Jim Rumbaugh OMG Unified Modeling Language Specification, 2003 An Adopted Formal Specification of the Object Management Group, Inc.

5 Abdelouahed Gherbi and Ferhat Khendek UML Profiles for Real-Time Systems and their Applications, J. of Object Technology 5:149-169 2006

6 OMG Object Management Group UML® Profile For Schedulability, Performance, And Time, Version 1.1, 2005

7 B. Selic and J. Rumbaugh. Using UML for Modeling Complex Real-Time Systems. March 1998 White-paper p. 1-22, and Branislav Selic Using UML for modeling complex real-time systems, LNCS Languages, Compilers, and Tools for Embedded Systems 1474:250-260 1998

8 B. Selic, G. Gullekson, and P.T. Ward. Real-Time Object-Oriented Modeling. John Wiley and Sons, 1994.

9 UML Profile for Modeling QoS and FT Characteristics and Mechanisms Specification, v1.1, 2008 p. 1-92 OMG Object Management Group

10 http://www-01.ibm.com/software/rational/announce/rose/?S_TACT=105AGX23&S_CMP=ROSE (now from IBM, accessed Okt. 2009)

11 R. Grosu, M. Broy, B. Selic, Gh. Stefanescuz Towards a Calculus for UML-RT Specifications, Object-Oriented Programming Systems, Languages and Applications, OOPSLA, 1998

12 J. C. P. Woodcock and A. L. C. Cavalcanti The semantics of circus, LNCS 2272:184-203 2002

13 Rodrigo Ramos, Augusto Sampaio, Alexandre Mota Rigorous Development with UML-RT, 19th Brazilian Contest on Dissertations and Thesis (CTD'06), SBC 2006 p. 1-5

14 Michael von der Beeck A Formal Semantics of UML-RT, LNCS: Model Driven Engineering Languages and Systems 4199:768-782 2006

15 G. D. Plotkin A Structural Approach to Operational Semantics, DAIMI FN-19 1981 p. 1-132 University of Aarhus

16 Zonghua Gu and Zhimin He Real-Time Scheduling Techniques for Implementation Synthesis from Component-Based Software Models, Lecture Notes in Computer Science: Component-Based Software Engineering 3489:235-250 2005

17 Z. Gu and K. G. Shin Synthesis of Real-Time Implementation from UML-RT Models, LNCS: Component-Based Software Engineering 3489:235-250 2005

18 Jochen M. Küster and Joachim Stroop Consistent Design of Embedded Real-time Systems with UML-RT, Object-Oriented Real-Time Distributed Computing, 2001 p. 31-40 IEEE

19 Paul Whittaker, Michael Goldsmith, Kirk Macolini, Tim Teitelbaum Model Checking UML-RT Protocols., Workshop Formal Design Techniques for Real-Time UML 2000

20 Formal Systems http://www.fsel.com/

21 A. W. Roscoe Model-checking CSP in A classical mind: essays in honour of C. A. R. Hoare p. 353 - 378 1994

22 Ingolf Krüger, Wolfgang Prenninger, Robert Sandner Deriving Architectural Prototypes for a Broadcasting System using UML-RT, Eds. P. Kruchten, Proceedings 1st ICSE Workshop on Describing Software Architecture with UML 2001 Rational Software

23 Werner Damm, Bernhard Josko, Amir Pnueli, Angelika Votintseva A discrete-time UML semantics for concurrency and communication in safety-critical applications, Science of Computer Programming 55:81–115 2005

24 XML Metadata Interchange: http://www.omg.org/technology/documents/formal/xmi.htm

25 B. Pagès at http://bouml.free.fr

26 http://www.omnetpp.org

27 András Varga, Rudolf Hornig AN OVERVIEW OF THE OMNeT++ SIMULATION ENVIRONMENT, 1st International Workshop on OMNeT++ 2008

28 The integrated development environment eclipse also has an UML-modeling facility at http://www.eclipse.org/modeling/mdt/?project=uml2