Self-Adaptation in Geotracking Applications: Challenges, Opportunities and Models

Olga Melekhova, Mohammed-Amine Abchir, Pierre Châtel, Jacques Malenfant, Isis Truck, Anna Pappa

Self-Adaptation in Geotracking Applications: Challenges, Opportunities and Models

Olga Melekhova*, Mohammed-Amine Abchir†‡, Pierre Châtel†, Jacques Malenfant*, Isis Truck† and Anna Pappa†
*Université Pierre et Marie Curie-Paris 6 CNRS, UMR 7606 LIP6
4 place Jussieu, Paris, 75005, France
Email: {Olga.Melekhova,Jacques.Malenfant}@lip6.fr
†LIASD – EA 4383, Université Paris 8
2 rue de la Liberté, Saint-Denis Cedex, 93526, France
Email: {maa,truck,ap}@ai.univ-paris8.fr
‡Deveryware 43 rue Taitbout, 75009 Paris, France

***Abstract*—Geotracking emerges as a mission-critical mean to observe, track and otherwise manage mobile entities from the regular notification of their geographic locations, thanks to positioning devices such as GPS and cell phones. First used in logistic systems, geotracking now strives for large-scale with new applications like the management continent-wide green tax collection on millions of trucks from their passage through thousands virtual toll points. As the current single device, fixed position notification frequency policies can no longer cope with the stringent requirements of such large-scale applications, this paper first examines the challenges and opportunities for self-adaptation in geotracking systems and applications. After identifying the main requirements to that end, it then provides for a first set of decision-making models for identified self-adaptation *scenarii*, in the context of a set of everyday geotracking business-oriented applications. Finally, it proposes a set of tools and the software architecture currently developed under the French ANR project SALTY to address these issues in the conceptual framework of autonomic computing.**

***Keywords*-Adaptive systems; Road vehicle location monitoring; Closed loop systems; Decision-making; Fuzzy control; Global Positioning System; Man-machine interface; Natural language interfaces; Software architecture.**

I. INTRODUCTION

As geotracking and its use in geographic information systems are more and more prevalent, approaches to optimize their resource usage for given business objectives become a key issue. Typical geotracking applications use periodic position notification from remote positioning devices to track mobile targets. The goal is to follow their progress and enforce geographic constraints such as staying within a predefined corridor or passing through waypoints.

Current practices in geotracking address all of the business objectives using predefined fixed frequencies for devices to push new positions. As each push incurs a cost in data transmission and in device battery usage, fixed frequencies impose quite poor resource usage profiles. Moreover, new programmable positioning devices offer much more flexibility than simply applying a fixed notification frequency. Finally, as geotracking go large-scale, with new applications like the management of continent-wide green tax collection on millions of trucks from their passage through thousands of virtual toll points, striving for adaptive policies and coping with device malfunctions become mandatory.

This paper presents results of the French ANR project SALTY[1] by first exploring the challenges and opportunities in the self-adaptation of geotracking applications. After presenting the conceptual framework underlying geotracking and its self-adaptiveness (§II), the paper takes a bottom-up approach by describing in Section III self-adaptation use cases for a set of exemplary business-oriented geotracking goals: corridor enforcement, waypoint passage notification and delivery point arrival notification. Section IV further develops over the adaptation needs by providing decision models specific to these use cases but general enough to cope with most adaptation requirements in geotracking applications. Requirements and first drafts for a software architecture and accompanying tools are described in Section V.

II. GEOTRACKING AND SELF-ADAPTATION

Self-adaptation in geotracking strives for a minimal resource usage while sustaining the capability for the system to achieve business objectives, like enforcing a corridor within which a mobile must stay, at some desired level of precision, *e.g.*, notifying the crossing of a corridor frontier with a precision of 500 meters. To better understand how this overall goal can be achieved, this section examines the devices, software platforms and kinds of adaptation that exist in this area.

A. Positioning devices

Geotracking relies on positioning devices which characteristics deeply impact costs and resource usage. Positioning devices fall into three categories of positioning means:

1) Global Positioning System (GPS) devices are now well-known; they use information and triangulation of signals coming from satellite constellations. Their precision highly depends upon the number of satellites they can "see", the angle between them and even the ground configuration which can cause some bias. After computing

[1]Self-Adaptive very Large disTributed sYstems (ANR-09-SEGI-012).

 ISBN: 978-1-61208-109-0

positions, devices typically push them to a server using data connection over a GSM network.

2) GSM cell-id uses readily available phone (or similar devices connected to that network) tracking information, *i.e.*, the cell in which the device actually is, to provide estimated positions. Its precision highly depends upon the density of antennas and the geographical form of cell in which the phone is. It needs no specific intervention from the phone itself but is rather a service offered by the cell phone infrastructure. Upon a call to their API, operators provide estimated positions from the geographical locus of the cell containing the phone.
3) WiFi cell-id is similar to GSM cell-id but rather uses WiFi antennas to estimate positions of devices connected to that network. Again, the precision of the method depends upon the density of the antennas.

Devices can be fixed or portable. Indeed, specific GPS devices can be attached to vehicles and be powered by them. Such devices are usually also connected to sensors on the vehicle (door open/closed, motor temperature and speed, fuel level, ...) and can send sensor data along with positions. But more and more portable GPS devices are used for their flexibility, at the expense of running on batteries, which then become a scarce resource. Mobile phones share characteristics with portable GPS, while WiFi cell-id is totally dependent on the device to which the WiFi card is connected.

Most GPS devices send positioning (and possibly other) data through a data link over the GSM network at a frequency that can be dynamically modified by sending them an appropriate command. Many devices can be partly or totally put in sleep mode for economizing energy: the communication subsystem (GSM) can be put in sleep mode between data sendings while the positioning subsystem itself can also be put in sleep mode between signal acquisition and position computations. More and more programmable devices appear everyday on the market. Such devices can embed application specific code that can drive the positioning but also adaptations of this process to some extent.

B. Geotracking concepts and middleware

One of the key issue in building geotracking applications is how to cope with the diversity of positioning devices. No standard interface exists neither to get positions from devices nor to send them commands. Instead of making end user applications aware of all the possible devices, and have them track down every new devices to be used, a middleware layer can help in abstracting them from this diversity. Deveryware's GeoHub is one such middleware that offers, among other services, a common interface to deal with a large set of different positioning device kinds and models.

Another important issue is the capability to acquire a lot of positioning data and to correlate them in order to capture higher level composite events triggering notifications to end user's applications. Falling under the framework of complex event processing (CEP) [1], indeed geotracking need specialized CEP engines coping specifically with positions and related events. Deveryware's GeoHub central role is to act as such a specialized geotracking CEP, which triggering rules are written in Forth, uploaded to the hub and activated over period of times at customer demand. GeoHub optimizes the process by which positions received from tens of thousands of positioning devices fire end users' notification rules linked to their distant applications.

C. Potential adaptations

As the basic and prominent functioning of positioning devices consists in sending data at some interval of time, the frequency at which a device must do so can be determined directly from business objectives and the precision required by end users in their notifications.

Example. Consider the notification of the passage of a vehicle within 500 meters of a waypoint. If the speed of the vehicle is 100 kph at this point, the frequency must be at least one data sent each 36 seconds to make sure a position will be taken within the 1 km diameter around the waypoint.

But, maintaining such high frequencies while vehicles are far away from any point of interest is just a loss of resources and unnecessary costs. A central adaptation in geotracking is therefore to adjust the frequencies of data sendings so to keep them low when mobiles are far from points of interest but to raise them when approaching such points so to reach the required frequencies to match business objectives with the desired precision. Unfortunately, determining when to raise the frequency requires to predict the position of the vehicle from time to time, to get gradually nearer positions to the objective in order to raise the frequency progressively. Such position predictions depends on a lot of dynamic conditions, like weather, road conditions, traffic, etc. Therefore, dynamic adaptation is mandatory.

Besides this, another important issue is to cope with imprecision or unavailability of devices. GPS positioning precision depends upon the visible satellites and even the ground configuration, which can induce systematic bias, while other means highly depend upon the density of antennas. Switching from one positioning mean to another when conditions require it is another very important adaptation opportunity. As costs of the different mean vary much, GSM cell-id being typically a pay-per-use service from cell phone network operators, keeping their use to a minimum is also an adaptation objective.

Finally, for devices which autonomy on battery is limited, adaptation shall put the device positioning and transmission subsystems in sleep or ready modes back and forth. Putting the transmission subsystem in sleep mode while keeping the positioning subsystem in ready mode taking positions at some frequency can give rise to a form of location-driven positioning instead of the standard time-driven one. Programmable devices can monitor their position and resume sending data only when reaching some target position (or use a fallback rule if the target is missed).

In the SALTY project context, another important adaptation comes from the management of the GeoHub resources. Even

 ISBN: 978-1-61208-109-0

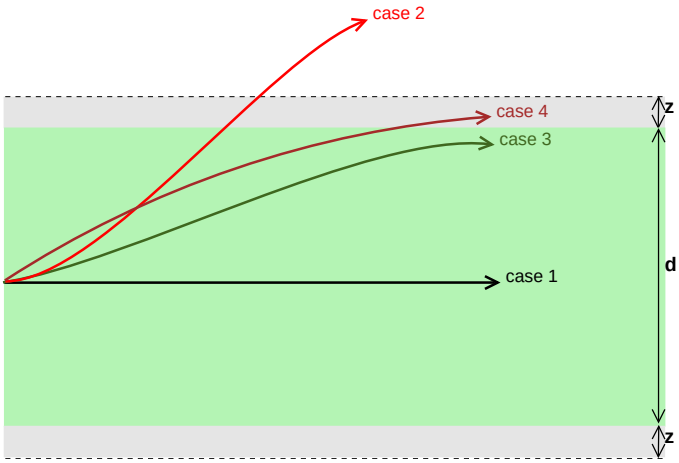

Fig. 1. Different behaviors in corridor enforcement.

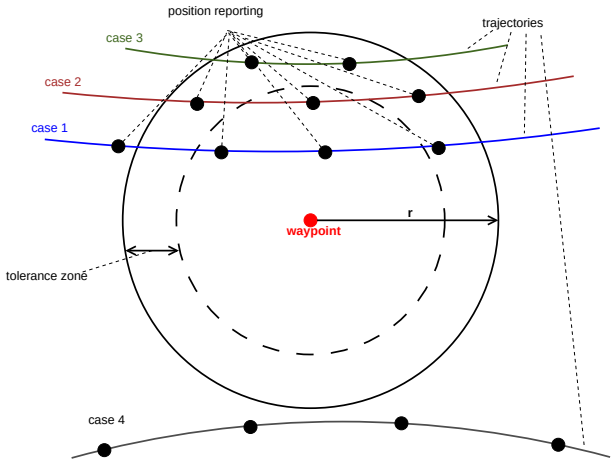

Fig. 2. Waypoint notification tolerance and mobile trajectories.

though more servers can be dynamically allocated to the GeoHub, variations in the workload can still require to adjust more globally the frequencies of mobile data sendings to keep the quality of service in position correlations done by the platform. As the workload of the GeoHub is the sum of the ones generated by every connected device, this kind of adaptation requires to act upon a lot of devices and therefore must scale well when their number grows larger and larger.

III. Business objectives and adaptation use cases

Geotracking applies to a lot of different areas, such as logistics, people tracking, traffic monitoring, etc. However, Deveryware's experience shows that it revolves around a limited number of similar high level business objectives. This section illustrates them within the context of logistic applications.

A. Corridor enforcement

A first typical geotracking objective is corridor enforcement, where positions of the vehicle must remain within a short distance from its planned route. End users require to be notified whenever the vehicle goes out of this limit, with a given precision. Typically, the corridor may be d = 10 km wide, and the end user may require a notification of the crossing of a frontier with a precision of z = 500 meters. This can trigger simple route replanning if, after verification, the vehicle happens to try to escape traffic jams, but it can also trigger a more urgent intervention if the vehicle is carrying valuable goods that may just being stolen.

Figure 1 shows in more details *scenarii* in corridor enforcement, where the corridor itself is the green area. The nominal scenario (case 1) is when the vehicle remains more or less in the middle of the corridor, a case where the frequency can be kept relatively low, given the width of the corridor. A second scenario (case 2) is when the vehicle crosses the frontier. The frequency is raised as the vehicle approaches the frontier, and then the crossing of the corridor frontier is notified. The end user may require a tracking of the vehicle after the notification, but typically with less precision, so the frequency may then diminish to adapt to this new requirement

A third scenario (case 3) is when the vehicle approaches a frontier of the corridor, but remains inside it. In this case, a higher frequency will be required to make sure to notify the user with its 500 meters (= z) precision if crossing occurs, and it will be kept as long as the vehicle remains in this situation.

In the last scenario (case 4), the vehicle crosses the frontier but remains within the 500 meters limit outside the corridor. The business rule here says to tolerate these positions, but as it is not nominal, a notification is required if this situation stands still for a too long time (or more relevantly, a combined measure of distance to the frontier and time spent in that situation). Again, the frequency must increase to notify a possible full crossing, or the reaching of the tolerance limit, and maintained as long as this situation persists.

B. Waypoint passage notification

A second typical geotracking objective is waypoint passage notification. In logistics, waypoints represent meaningful steps when fleet managers want to get feedback to make sure vehicles progress normally on their route, or inquire the drivers if not. In other applications, waypoint notification may serve as basis for green tax collection, as discussed previously.

Figure 2 shows the different *scenarii*. If waypoints are used only to check the progress of the vehicle, the notification may just require to get a position within some radius of the waypoint, *e.g.*, 500 meters, with a tolerance of 100 meters. In this case, it suffices to raise the frequency to a level where we can guarantee that a position will be measured within the circle, at least for a trajectory which passes near the middle of the circle (case 1). Indeed, a higher frequency may be needed for trajectories that will just cut a small part of the circle, as we can see in the case 3, that a fixed frequency will not provide the necessary positions. Overall, the frequency will need to be raised as the vehicle approaches the circle, and maybe to a higher level if the predicted trajectory cuts less of the circle.

 ISBN: 978-1-61208-109-0

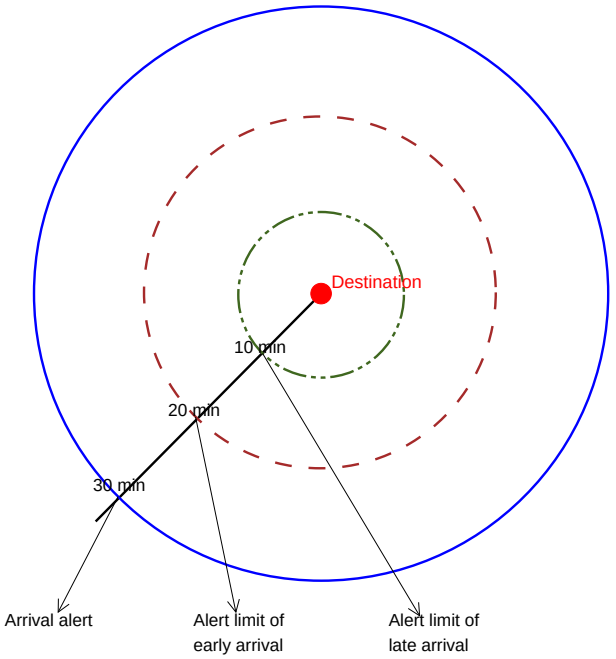

Fig. 3. Delivery point: first notification, early and delayed arrival.

A more stringent scenario comes with applications like the green tax collection. Here, owners of tracked vehicles may require more than one position to be sure the vehicle really passed through the waypoint, to accept to pay the tax. The rule may require three positions within the circle with at most one in the tolerance zone. In such a case, the frequency must be raised much when approaching the circle with a more or less tangent trajectory. In this scenario, being the resource provider for data sendings, the taxation authority may have to balance the cost of raising the frequency against the revenue generated by the tax payment, and decide at some point to give up if the trajectory is as tangent as the case 3.

C. Delivery point arrival notification

The delivery point arrival notification differs from the waypoint passage by the fact that the vehicle will eventually stop at the destination point. Hence the problem is not to detect a passage but rather to notify the arrival point of the estimated time at which the vehicle must be expected. In logistic applications, such a notification, *e.g.*, 30 minutes in advance, will be used by warehouse dispatchers to choose a door at which the vehicle will be able to stop, deliver (part of) its payload, and take some other payload. As warehouse doors are scarce resources and vehicles and drivers wait time to be minimized (especially when the goods are perishable), a good planning with as much information as possible about incoming vehicles may be a key to success.

Figure 3 illustrates this third business-oriented objective. Typically, a first circle is defined to notify the warehouse of the arrival of the truck. But if the vehicle turns out to progress faster than predicted, an early arrival notification may be required no later than a second limit, *e.g.*, 20 minutes before the predicted time of arrival. Alternatively, if the vehicle is slower than predicted, a late arrival notification may also be required not later than another limit, *e.g.*, 10 minutes before the predicted arrival time.

Adaptation of the frequency follows here a more complex profile. It raises as the vehicle approaches the first limit, in order to notify the arrival with some fixed precision, *e.g.*, 30 minutes more or less 2 minutes. After this first notification, the frequency is reduced, but if the vehicle appears to be too fast or too slow, the frequency is again raised to catch the other time limits with their own predefined tolerance.

D. Mixed scenarii and other adaptations

Of course, in logistic applications, the three business objectives can be active at the same time. For example, the corridor may still need to be enforced while passing a waypoint or approaching a destination. Adaptation may therefore need to take care of several objectives at the same time and decide for a frequency that matches all of their needs. Other types of adaptations may also be used or may constrain these. For example, putting the device in sleep mode for a while when no objectives are in sight can preserve the battery for a better usage later. Adapting location-driven data sendings discussed before may also be used if programmable devices are available, at least to come to some point nearby the zone of interest. Finally, if problems are detected in the precision of the positions or a device suffers from malfunction, switching to an alternative positioning device may be required.

IV. Adaptation Model and Decision Making

This section introduces the adaptation model capturing the dynamic of the system in terms of states, decisions, transitions and cost functions to set up de the decision-making problem to be solved to get adaptation policies.

A. Positioning devices adaptation model

Let D be the set of device types and P be the set of GPS positions, the state of a vehicle can be described by the following state variables:

- $pd \in D$: the type of the primary device used by the truck,
- $sd \in D$: the type of the secondary device used by the truck,
- $ct \in \mathbb{R}$: current time,
- $cp \in P$: the current position of the truck,
- $cf \in \mathbb{R}$; the current frequency of data sendings,
- $bl \in \mathbb{R}$: current battery level,
- $mm \in \{sleep, on\}$: the current measurement unit mode,
- $tm \in \{sleep, on\}$: the current transmission unit mode,
- $sp \in \{ok, failed\}$: the state of the primary device,
- $ss \in \{ok, failed\}$: the state of the secondary device,
- c: the corridor imposed to the vehicle,
- w: the list of the remaining waypoints,
- a: the list of the remaining warehouses to visit.

Each time a decision must be made, this decision will take the form of three components:

 ISBN: 978-1-61208-109-0

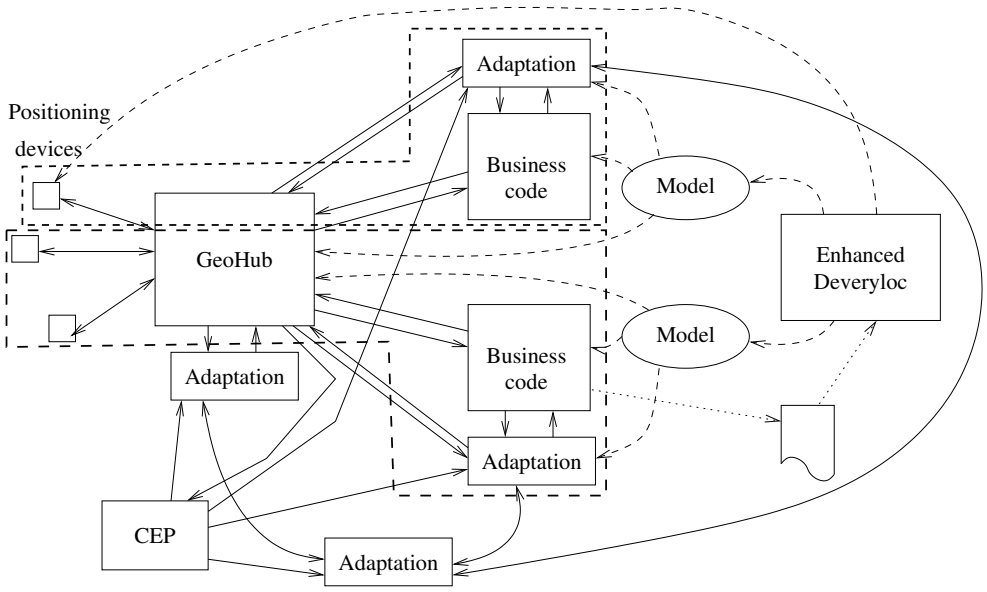

Fig. 4. Elements of the logical architecture.

- $d \in \{primary, secondary\}$: the device that will be used for the data sendings until the next decision,
- $f \in \mathbb{R}$: the frequency at which data will be sent until the next decision,
- $t \in \mathbb{R}$: the time until the next decision, if no failure, battery drained, corridor, waypoints or warehouse arrival event occur.

The cost of transmission is modeled by a function $c_{dt}(t, f)$ giving the total cost of data transmission for a device of type/model dt, a time duration t and a steady frequency f over that period. For GSM cell-id, this cost turns out to be the service billed by the operator for calling $n = t \times f$ its positioning API. For GPS devices, this is the cost of transmitting n packets of data over the GSM network.

Energy consumption for battery-powered devices is modeled by a function $e_{dt}(m, m', t, f)$ where

- dt is the type/model of the positioning device,
- m, m' tells if either the measurement or the transmission subsystems are to be put in the new mode m' from a previous mode m,
- t is the elapsed time, and
- f is the frequency of measurement and transmission of data (only measurement if the transmission subsystem is in sleep mode).

Transitions to new states are modeled using conditional probability distributions, where cp is the current position:

- $P_c(t|c, cp)$: the probability that the next position in t unit of time will have passed over corridor c,
- $P_w(t|w, cp)$: the probability that the next position in t unit of time will have passed the next waypoint in w,
- $P_a(t|a, cp)$: the probability that the next position in t unit of time will have passed the next warehouse arrival notification in a,
- $P_d(t)$: probability that a failure will occur on a device $d \in \{primary, secondary\}$.

Modeled as above, the goal of the decision-making process is to find an adaptation policy that provides a decision (d, f, t) for each state s, which minimizes the cost of data sendings and the probability of a miss of a corridor crossing, waypoint miss or of an unnotified warehouse. Two kinds of algorithms can be applied to compute optimal decision policies: sequential decision making algorithms and greedy algorithms.

When decisions influence each others over time, such as putting a device in sleep mode from which an awakening cost will have to be amortized over the sleep time, algorithms for sequential decision making are required. When the model (cost functions, transition probabilities, and so on) is known in advance, such markovian decision processes can be solved using dynamic programming [2], [3], yielding a policy in the form of a function from states to decisions. Such policies can then be applied at run-time at low cost (in decision making time). When the model elements are not known in advance, run-time machine learning algorithms like Q-Learning [4], [3], are applied.

When decisions do not influence each others over time, greedy, "one-step", algorithms can be applied. The field of optimization offers a lot of such algorithms. As most of the time, the model elements are not known in advance, approaches amenable to run-time learning are preferred. Fuzzy control, introduced by Zadeh [5], proposes such an approach [6].

B. GeoHub adaptation model

As said earlier, if the GeoHub can have its resource augmented to cope with higher workload, these additions of resource come in such large increments that the level of data sendings to it must be controlled to avoid too large decreases in

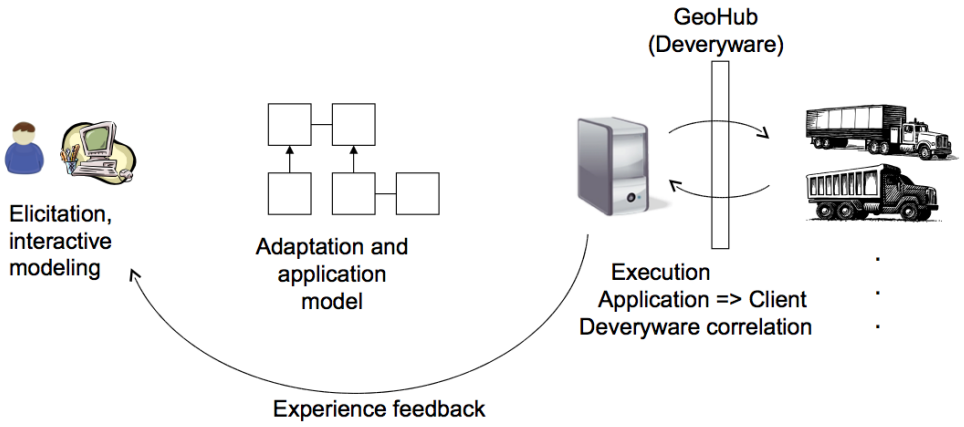

Fig. 5. Overall development steps for geotracking applications.

the quality of service offered by the platform between resource allocations. Controlling the workload of the GeoHub amounts to the control of the frequency of each positioning device currently sending it data. Moreover, as the SALTY project addresses the large-scale, this must be done in a way that scales to a large number of positioning devices. We therefore propose to use decentralized, peer-to-peer algorithms to do so.

When the workload of the GeoHub becomes too large, two kinds of algorithms can regulate it:

- flooding algorithms, which is the diffusion of a command from the adaptation component of the GeoHub to all autonomic managers of devices to reduce their frequency in order to reduce its overall workload, or
- token-based algorithms, where the adaptation component of the GeoHub injects to the autonomic managers of devices tokens representing rights to use some frequency to send data, tokens which will be used and then reinjected towards others when no longer needed.

In the token-based algorithm, a budget of tokens is given to each autonomic manager of positioning devices, which then raises and drops the frequency of its devices within these budget limits. When some autonomic manager requires more tokens because targets approach their next point of interest, thus requiring higher frequencies, a peer-to-peer heuristic algorithm is applied: autonomic managers ask their neighbors, and then these again recursively until one with less current needs than its current budget can give some tokens for a period of time.

V. Software Architecture and Tools

In the context of the SALTY project, use cases in logistics are developed into an overall logical architecture, shown in Figure 4, and described in the rest of this section.

A. Application architecture

Applications include business-oriented code with regards to the geotracking objectives and corresponding reactions from a business point of view. For example, this code will be in charge of enforcing a corridor over the route of a vehicle, and if the vehicle exits the corridor, business rules will encode the instructions of the end user to deal with this situation, like calling the driver to know why he did so, and replan a new route if this deviation is justified (traffic jams, accidents, ...).

But applications also include the notification logic and its adaptation. In Figure 4, applications are materialized by bold dashed polylines, which include the business code discussed above, but also the adaptation components. The geotracking notification logic is also part of the application though delegated to the GeoHub and positioning devices. In this vision, the code on programmable devices and the rules loaded into the GeoHub are seen as part of the application, and it is the decision of the programmer as where to put the frontier between what is delegated to these and what will be executed on its own infrastructure. It is one of the objective of the SALTY project to better use programmable devices that appear on the market regularly and promise to lower the costs of geotracking, provided that they are properly used.

Not shown in the figure is the organization of the business code layer, for the sake of understandability. In typical applications, several components will be used to cope with the different business objectives. Typically, components are specialized to specific geotracking objectives, like corridor enforcement. In terms of deployment, each vehicle is usually followed individually, so having one set of specialized components per vehicle is typical of this kind of applications.

B. Local autonomic management architecture

Each application has its own adaptation layer, which implements the adaptation logic and preferences of the end user. For example, one may put more importance on meeting its objective with a high precision, while another may prefer to keep the overall geotracking costs as low as possible. The basic logical architecture of the adaptation layer follows IBM's autonomic computing blueprint and its Model - Analyse -

 ISBN: 978-1-61208-109-0

```
<?xml version="1.0" encoding="iso-8859-1"?>
<geotracking-model
    uri="http://www.deveryware.com/model/TruckTracking"
    xmlns="http://move.lip6.upmc.fr/geotracking-model/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <alertType name="ArrivalAtWarehouseNotification"
            category="ArrivalAtADestination">
  <application-parameters>
   <param name="Delay" type="xsd:time"/>
   <param name="DelayTolerance"
          type="QualitativeTimeDelayTolerance"/>
  </application-parameters>
  <configuration-parameters>
   <param name="PositioningDevice"
          type="PositionDeviceType"/>
   <param name="StartTime" type="xsd:dateTime"/>
   <param name="EndTime" type="xsd:dateTime"/>
  </configuration-parameters>
  <runtime-parameters>
   <param name="CurrentPosition" type="GPSValue"/>
   <param name="WarehouseLocation" type="GPSValue"/>
  </runtime-parameters>
  <trigger type="FIS">
   <param name="Pos" type="GPSValue"/>
   <fis name="triggerArrivalNotification">
    <fuzzyRule>
     <antecedent type="QualitativeTimeTolerance">
      <two-tuple linguistic-variable="closeTo"
                 displacement="0.0"/>
     </antecedent>
     <consequent type="TriggeringLevel">
      <two-tuple linguistic-variable="High"
                 displacement="0.25"/>
     </consequent>
    </fuzzyRule>
    ...
   </fis>
   <infer fisName="triggerArrivalNotification">
    <expression op="minus">
     <applyFunction name="estimateTravelTime">
      <with-param name="from"
                  value="$CurrentPosition"/>
      <with-param name="to"
                  value="$WarehouseLocation"/>
     </applyFunction>
     <value-of exp="$Delay"/>
    </expression>
   </infer>
  </trigger>
  <emittedEvent
     type="ArrivalAtWarehouseNotificationEvent"/>
  <description lang="en-En">
   When a truck can be predicted to arrive at the
   specified warehouse at the given Delay, the
   specified event is emitted towards the
   application.
  </description>
 </alertType>
 <dataType name="QualitativeTimeTolerance">
  <fuzzyEnumeration>
   <linguistic-variable name="closeTo">
    ...
   </linguistic-variable>
   ...
  </fuzzyEnumeration>
 </dataType>
 <dataType name="TriggeringLevel">
  <fuzzyEnumeration>
   ...
   <linguistic-variable name="High">
    ...
   </linguistic-variable>
   ...
  </fuzzyEnumeration>
 </dataType>
 <dataType
     name="ArrivalAtWarehouseNotificationEvent">
  <data name="TruckId" type="xsd:token"/>
  <data name="occurenceTime" type="xsd:dateTime"/>
  <data name="WarehouseLocation" type="GPSValue"/>
 </dataType>
</geotracking-application>
```

Fig. 6. Models captured through the new DeveryLoc tool in XML format.

Plan - Execute (MAPE) loop [7]. The "monitor" part gets the notifications and positions from the GeoHub, but can also get information about the context from a standard CEP (weather, traffic, etc.). The "analyze" part deals with the decision model presented in Section IV. The "plan" part looks at how to implement the decision, and does the necessary coordination with other related adaptation components. Finally, the "execute" part deals with the actual adaptations acting not only upon the business code component but also on positioning devices and on the GeoHub when required.

To this logical architecture corresponds an implementation architecture, which makes use of several different adaptation components. As several decision making processes apply (adapting frequencies, choosing the positioning mean/device, switching from ready to sleep modes back and forth, etc.), instead of having a unique adaptation component dealing with all of these at the same time, several such components can be implemented, one for each decision problem. These components are then assembled together with an arbitrator composing their independently made decisions into one coherent decision that will be imposed to the base business code component as well as the concerned positioning devices and GeoHub.

Local coordination among tightly-coupled adaptation components is often required as, in general, adaptations have impacts on more than one base element. It is the case here for adaptation components that relate to the specialized geotracking components in charge of one particular vehicle, as they all use the same positioning device. For such coordination among a small number of adaptation components, they can simply be fully interconnected. When the number of components grows, a hierarchical scheme may help in implementing appropriate coordination algorithms.

C. Global coordination architecture

In some cases, adaptations need to be done at a much larger scale, and even globally. In the architecture of Figure 4, the adaptation component linked to the GeoHub controls its overall workload and performance, as explained in Section IV-B. To support the kind of peer-to-peer coordination algorithms required to go large-scale, the architecture links together the adaptation components managing each positioning device in

 ISBN: 978-1-61208-109-0

```
<?xml version="1.0" encoding="iso-8859-1"?>
<adaptation-model
   uri="http://www.deveryware.com/model/TTAdaptations"
   xmlns="http://move.lip6.upmc.fr/adaptation-model/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <adaptationType name="TransmissionFrequencyAdaptation">
  <state name="currentPosition" type="GPSValue"/>
  <state name="transmissionFrequency"
         type="xsd:float"/>
  <action name="newTransmissionFrequency"
          type="xsd:float"/>
  <criterion name="getPositionProbability"
             relation-semantics="increasing">
   <function>
    <param name="targetPosition" type="GPSValue"/>
    <param name="targetRadius" type="xsd:decimal"/>
    <expression>
     if (withinRadius(nextPosition(currentPosition,
                         newTransmissionFrequency),
                      targetPosition, targetRadius))
     then 1
     else if (beforeRadius(predictedPosition(
                         currentPosition,
                         newTransmissionFrequency),
                      targetPosition, targetRadius))
     then 1
     else 0
    </expression>
   </function>
  </criterion>
  <criterion name="batteryUsage"
             relation-semantics="decreasing">
   <function>
    <param name="consumptionPerTransmission"
           type="xsd:float"/>
    <expression>
     <value-of exp="$percentagePerTransmission"/>
    </expression>
   </function>
  </criterion>
  <criterion name="totalTransmissionCost"
             relation-semantics="decreasing">
   <function>
    <param name="costPerTransmission"
           type="xsd:float"/>
    <expression>
     value-of exp="$costPerTransmission"/>
    </expression>
   </function>
  </criterion>
  <aggregation>
   <function>
    <param name="getPositionProbability"
           type="xsd:float"/>
    <param name="consumptionPerTransmission"
           type="xsd:float"/>
    <param name="costPerTransmission"
           type="xsd:float"/>
    <expression>
     ...
    </expression>
   </function>
  </aggregation>
 </adaptationType>
</adaptation-model>
```

Fig. 7. Adaptation models captured through the new DeveryLoc tool in XML format.

a self-adaptive mesh striving for an equilibrium between the number of neighbors and the average length of the path connecting any randomly chosen pair of components.

With this kind of mesh in place and correctly maintained by the system, the architecture will efficiently support several kinds of peer-to-peer coordination algorithms:

- flooding algorithms, where the diffusion of a command can be done in time linear in the average path length, or
- token-based algorithms, where the time between the release of a token and its acquisition by another device is also linear in the average path length.

In the context of the SALTY project, the self-adaptive mesh is constructed using heuristic probabilistic algorithms. When a new autonomic component connect its positioning device to the GeoHub, the latter's autonomic component assigns neighbors to the newcomer at random among the already connected ones. Then, these can exchange neighbors with their neighbors if they tend to have more than them. Precise constraints to make this kind of heuristic construction converge to the above good equilibrium are still under study.

D. Application and decision-making model elicitation

The last part of Figure 4, on the right, presents a software tool that completes the logical architecture with an aid for end users to specify their application and eventually connect it to the overall system. Deveryware already offers such a service, called DeveryLoc, which helps end users to configure alerts of predefined types, that is both the triggers and the corresponding notifications, load them onto the GeoHub and start/stop them at will. Currently, this development tool still requires a lot of expertise in geotracking that is rarely available to the clients. Another way to help end users is to propose smart human-machine interfaces to help them specifying their needs. So, the new enhanced DeveryLoc has four major objectives:

1) use the natural language (more precisely a basic discourse analysis) to enhance the HMI in proposing "user-understanding" interfaces;
2) elicit the business-level objectives and their parameters, such as corridor enforcement;
3) elicit the adaptation objectives, such as keeping the frequency low, to save battery or to get a required level of precision in the notifications;
4) enable a qualitative assessment of criteria, instead of a quantitative one, often less intuitive for end users.

The overall needs of the clients are expressed through four different artifacts:

1) an application model defining the kinds of alert types found in the application domain, such as arrivals at warehouse in truck tracking (Fig. 6);
2) an adaptation model defining the kinds of adaptations that can be applied to alerts, such as a position transmission frequency adaptation (Fig. 7);

 ISBN: 978-1-61208-109-0

```
<?xml version="1.0" encoding="iso-8859-1"?>
<geotracking-application uri="http://www.deveryware.com/application/TruckTrackingApp"
                         applicationModel="http://www.deveryware.com/model/TruckTracking"
                         adaptationModel="http://www.deveryware.com/model/TTAdaptations"
                         xmlns="http://move.lip6.upmc.fr/geotracking-prog-model/"
                         xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <alert name="ArrivalAtNiceWN" type="ArrivalAtWarehouseNotification">
    <with-param name="WarehouseLocation"><gpsValue lat="43.7" long="7.26"/></with-param>
    <with-param name="Delay" value="00:30:00"/>
    <param name="DelayTolerance" value="closeTo"/>
  </alert>
  ...
  <adaptation name="frequencyForArrivalAdaptation" type="TransmissionFrequencyAdaptation"/>
  ...
</geotracking-application>
```

Fig. 8. Application description captured through the E-DeveryLoc tool.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<geotracking-configuration uri="http://www.deveryware.com/configuration/TruckTrackingConfig"
                           application="http://www.deveryware.com/application/TruckTracking"
                           type="http://www.deveryware.com/application/TruckTracking"
                           xmlns="http://move.lip6.upmc.fr/geotracking-model/"
                           xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <alertConfiguration alert="ArrivalAtNiceWarehouseNotification">
    <with-param name="PositioningDevice">
      <mobile name="truck1234" xmlns="http://www.deveryware.fr/schema/mobile"
        <phoneNumber>06123456</phoneNumber>
      </mobile>
    </with-param>
    <with-param name="StartTime" type="2010-10-26T08:00:00"/>
    <with-param name="EndTime" type="2010-10-31T18:00:00"/>
  </alertConfiguration>
  ...
</geotracking-configuration>
```

Fig. 9. Application configuration captured through the E-DeveryLoc tool.

3) an application defining alerts from the alert types of an application model to which it applies adaptations from the adaptation model (Fig. 8); and,
4) a configuration defining the exact devices and period of time during which the alerts of an application are activated on the GeoHub (Fig.9).

The separation of concerns exhibited by these different models allows us to define and give different roles in model elicitation. Geotracking specialists can define new application and adaptation models, while end users can define applications and configurations from libraries of existing models.

An example model from the logistics use case, elicited according to the above principles is shown in Figure 6. It corresponds to the models appearing on the right of Figure 4. A model defines alert types, which defines triggering conditions and the notification to be sent. Notice that alert type parameters are divided in three categories:

1) application parameters, that will be provided when defining an application,
2) configuration parameters, given at configuration time, and
3) runtime parameters provided by the GeoHub upon reception of positions to decide whether or not the alert is triggered.

To give a glimpse of the qualitative approach, the trigger of the alert is defined as a fuzzy inference system, based on the process of computing with words [8] here under the 2-tuple [9] representation model. In this case, this approach allows the end user to work with a qualitative tolerance for the time before the truck arrives at the position corresponding to the notification delay in order to trigger this notification. This makes more sense than a precise tolerance, like 2 minutes, given the imprecision of the time estimation made from positions and travel conditions of the truck.

Figure 7 shows the companion adaptation model capturing the kinds of adaptations to be applied to alert types. In this example, the adaptation model defines an adaptation of the frequency of position sendings. Adaptations identify the state information from the system upon which a decision must be made, and the actions that need to be taken. It also defines the criteria, and when there are several ones, an aggregation of these criteria must be fed into the decision process; one possibility is to define an aggregation function computing a single value from the previous criteria.

Figure 8 shows an example of an application. In this example, an alert of the type ArrivalAtWarehouseNotification is defined for the Nice warehouse by simply providing the application parameters to the corresponding alert type. Fig-

 ISBN: 978-1-61208-109-0

ure 9 then shows the configuration model which provides the configuration parameters for each and every alert in the corresponding application.

With this information, the tool shall generate code skeletons for both the business and the adaptation components, and also for the code to be loaded on the GeoHub and on the (programmable) positioning devices.

As it can be seen in Figure 4, and illustrated by Figure 5, a feedback link from applications to the new DeveryLoc will provide for offline *post-mortem* analysis of application *scenarii* to help end users tailor the parameters of their application needs and resources. For example, if it shows that the application has a too large cost, the end user may adopt less constraining tolerances on the notifications so to lower frequencies, and therefore the geotracking costs. Hence, it is adaptation at another level and under another time frame.

VI. Related work

The kinds of adaptations, as well as the described software architecture for geotracking, fall into the area of autonomic computing [7], an ambitious goal set by IBM in 2003 of building self-management capabilities in applications. Within this framework, our approach to adaptation falls under the self-configuration and self-optimization concepts, but also under the self-healing, though only using a fallback device.

GeoHub and the set of positioning devices used by the geotracking can be seen as a monitoring system. Hence, the kind of adaptation we have presented can be compared to adaptive monitoring [10], [11], but with dynamic adaptability capabilities [12]. Charbiwala et al. [13] address a problem similar to ours in the context of sensor networks, but they focus on correlation objectives rather than end users' application ones.

Within the field of autonomic computing, feedback control in distributed systems aims at applying control theory and tools to the adaptation of distributed systems, like the rate of sendings in sensor networks [13].

VII. Conclusion

In this paper, we have explored challenges and opportunities in the self-adaptation of geotracking applications. As geotracking applications go large-scale, a set of resource-aware decision-making models are proposed to strive for an optimal management of positioning devices, system-wide. Moreover, as these applications become more and more mission-critical, fault-tolerance is also being addressed, by integrating the possibility to switch between alternative positioning devices, like fixed GPS blackboxes placed on the vehicle, portable GPS devices running on batteries, and drivers' cell-phones, also running on batteries.

Switching to the geotracking system in the large, the paper also addresses issues in the management of a geotracking hub, used in SALTY to collect and correlate positioning data, as well as notifying end user applications when predefined conditions are met. The use case raises issues in the global management of a large set of positioning devices with regards to their joint use of a shared resource, and paves the way to large-scale coordination for fixing individual positioning frequencies so to minimize the resource usage system-wide, namely the rate of positions sendings coming to the GeoHub.

Having established these use cases and the required decision-making models, future work within the SALTY project include the design of a generic self-adaptive architecture based on the IBM's MAPE loop autonomic computing functional architecture. This generic architecture will be used in the implementation of demonstration applications for the SALTY project. Another important objective is to provide a methodology and a set of tools supporting the development of self-adaptive geotracking applications.

Acknowledgment

This work is partly funded by the French National Agency for Research (ANR), within the ARPEGE Program, under the project SALTY ANR-09-SEGI-012.

References

[1] S. Chakravarthy and Q. Jiang, *Stream Data Processing: A Quelity of Service Perspective*. Springer, 2009.

[2] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[3] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[4] C. Watkins and P. Dayan, "Q-Learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[5] L. Zadeh, "Outline of a new approach to the anamysis of complex systems and decision processes," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, no. 1, pp. 28–44, 1973.

[6] K. Michels, F. Klawonn, R. Kruse, and A. Nrnberger, Eds., *Fuzzy Control — Fundamentals, Stability and Design of Fuzzy Controllers*, ser. Studies in Fuzziness and Soft Computing. Springer-Verlag, 2006, no. 200.

[7] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[8] L. Zadeh, "The Concept of a Linguistic Variable and Its Applications to Approximate Reasoning," *Information Sciences, Part I, II, III*, vol. 8,8,9, pp. 199–249, 301–357, 43–80, 1975.

[9] F. Herrera and L. Martínez, "A 2-tuple fuzzy linguistic representation model for computing with words," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 6, pp. 746–752, 2000.

[10] M. A. Munawar and P. A. S. Ward, "Adaptive Monitoring in Enterprise Software Systems," in *SIGMETRICS 2006 Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML)*, 2006, http://research.microsoft.com/en-us/um/redmond/events/sysml/papers/sysml-Munawar.pdf.

[11] S. Agarwala, Y. Chen, D. Milojicic, and K. Schwan, "QMON: QoS- and Utility-Aware Monitoring in Enterprise Systems," in *Proceedings of the IEEE International Conference on Autonomic Computing, ICAC'06*. IEEE Computer Society, June 2006, pp. 124–133.

[12] B. Le Duc, P. Châtel, N. Rivierre, J. Malenfant, P. Collet, and I. Truck, "Non-functional data collection for adaptive business processes and decision making," in *Proceedings of the 4th Int. Work. on Middleware for Service Oriented Computing (MWSOC'09)*. ACM, 2009, pp. 7–12.

[13] Z. Charbiwala, S. Zahedi, and Y. Cho, "Toward Quality of Information Aware Rate Control for Sensor Networks," in *Proceedings of the FeBID 2009 Workshop*, 2009, http://controlofsystems.org/febid2009/papers/p11_s24_CharbiwalaZahedi.pdf.

 ISBN: 978-1-61208-109-0