GENSIMO — A Generic Framework for Modelling Social Insurance Systems

Fjalar J. de Haan[1,2][0000−0003−4715−0842] and Jason Thompson[3][0000−0002−3146−1198]

[1] School of Computing and Information Systems, The University of Melbourne, Australia
[2] Melbourne Centre for Data Science, The University of Melbourne, Australia
[3] Transport, Health and Urban Systems Research Lab, Melbourne School of Design, University of Melbourne, Australia
{fjalar.dehaan, jason.thompson}@unimelb.edu.au

Abstract. We report on the design, development and prototype testing of GENSIMO (GEneric Social Insurance Modeling), a free and open-source software framework for modelling and simulation of social insurance systems. We discuss the conceptual and software design considerations of the framework in general before demonstrating how it works in detail by setting it up to model a particular social insurance scheme. We then present how we used this prototype set up to model the impact of a policy intervention on the workload of the insurer. While GENSIMO is under active development and we are a while away from serious verification and validation, this prototype testing shows realistic results.

Keywords: Social insurance · Julia · Agent-Based Modelling · Free and Open Source Software

1 Introduction

We report on the design, development and prototype testing of GENSIMO (GEneric Social Insurance Modeling), a free and open-source software framework for modelling and simulation of social insurance systems. Typical examples of social insurance are workers' accident compensation schemes and unemployment benefits. We have a particular interest in road accident compensation schemes in Australia as this is our funding context (see the acknowledgements in Section 4). GENSIMO leverages the similarities amongst many schemes to provide a generic software framework that can be set up for, in principle, any scheme thus also enabling comparison. The software, under active development, is available as a Julia package under a GPLv3+ licence (`https://github.com/gensimo/Gensimo`) the current prototype is an application to the scheme of the Transport Accident Commission in Victoria, Australia (`https://www.tac.vic.gov.au/`).

We believe that modelling social insurance systems is societally important. Billions of public dollars are moved each year in schemes that touch millions of lives. Clearly, insurance organisations have not hitherto been going unmodelled.

Insurers typically have high-quality statistical and actuarial expertise, either in-house or outsourced. What we aim to provide with GENSIMO is a complementary approach that not only aids in forecasting workloads and liabilities but, in doing so, provides a representation of the workings of the insurance scheme. Our framework employs agent-based modelling (ABM) which allows us to directly model virtual clients interacting with a modelled scheme. Such a representation, we suggest, is an intuitive tool for policy experiments and scheme design.

Observing that simulation models, and in particular ABMs, are not the mainstream approach in the social insurance context. Thus we necessarily build from our own experience in this field and the broader ABM literature (which we assume the reader is familiar with). While GENSIMO's design and codebase are altogether new, we can draw on many years of experience developing models for social insurance systems. Of particular relevance are [4] and [3] which report on agent-based models for transport accidence insurance and workers compensation schemes, respectively. Most of our modelling work in this context is done in close collaboration with stakeholders which over the years has led to many lessons learnt, which were distilled in the form of guidelines for policy makers in [5].

A common feature in many social insurance schemes is that they are claim-based. Clients coming 'on scheme' have a claim with the insurer on the basis of which they get financial (typically) compensation for services they receive. For example, in an road trauma context, a client comes on scheme after an accident. Depending on the scheme, services the insurer might recompense include such varied things as surgery, physio therapy, income compensation and adjustments to living arrangements. Taking this common feature as the basis, GENSIMO essentially follows virtual clients on their journey through the scheme. Clients have a bipartite representation, with one part modelling their health status in the broadest sense, while the other part models their administrative status, i.e. their claim. On their journey, clients encounter events, such as undergoing assessments, requesting services and so on, each one of which may have consequences for their administrative and health states.

In the following, (Section 2) we present the design and development of the GENSIMO framework in general terms. After that, we report on its application to a particular insurance scheme as part of prototype testing (Section 3). There we compare two versions of a scheme to assess, amongst other things, impacts on workload. We end the paper with some brief conclusions (Section 4).

2 Design — Overview of the GENSIMO Framework

2.1 Software Design Considerations

Since our aim is to provide a generic framework it seemed obvious to use GENSIMO ought to be a downloadable, free and open source software package. And so it is, as you can verify at `https://github.com/gensimo/Gensimo` where it is available under a GNU General Public Licence Version 3+. We chose to develop

GENSIMO using the Julia language [1] as it is a fast languages extremely suitable for technical computing. In addition to the many useful facilities the language offers 'out of the box', we make extensive use of a number of Julia packages. Amongst these, the most important are:

Agents.jl This is Julia's main package for agent-based modelling [2]. It is easy to use and proclaims to be fast, simple and providing extensive tools. You can find it at `https://github.com/JuliaDynamics/Agents.jl`.

POMDPs.jl This is Julia's key package for modelling Markov Decision Processes (MDPs). It can be found here `http://juliapomdp.github.io/POMDPs.jl/latest/`. The GENSIMO framework also provides facilities to model a social insurance system as an MDP. This provides a convenient representation of the empirical data and this makes it suitable for validation as well. This aspect of the framework is not discussed further in this paper.

Later in this paper we will discuss how GENSIMO is set up to model a particular social insurance scheme. In software terms, this means that we have set up a separate Julia package (not publicly available as it would disclose sensitive information) that uses the GENSIMO package.

2.2 Backbone

The *backbone* of the GENSIMO framework is the client's journey through the scheme. When a client comes 'on scheme', the scheme begins to deploy processes involving that client. As a consequence, the client's recovery is helped or hindered, while the insurer incurs costs, labour and liability. GENSIMO models this with a `Client` data type that keeps track of physical and mental health (as well as other factors) and various administrative events. The backbone can thus be viewed a client state-changing machine.

Induction Client comes on scheme. Claim is opened.

Segmentation Client is allocated a service level based on health status or other personal parameters. A client may be re-segmented at later times.

Service request cycle Occurs zero or more times, depending on recovery trajectory.
- *Service request* — A product or service the client is seeking compensation for.
- *Service decision* — Process of deliberation culminating in the request being approved, denied or amended.
- *Iatrogenic effects* — The impact the service decision process has on the health and satisfaction of the client. E.g. psychological ramifications of repeatedly denied requests.

Recovery Background overall improvement of client's condition over time (exponential). GENSIMO does not model medical aspects in detail.

Inactivation After a period of inactivity a client is deemed 'inactive' or exits the scheme, depending on the insurer.

Fig. 1. GENSIMO backbone flowchart of processes in quasi sequence.

The processes in the backbone can be summarised in a quasi-sequential scheme as like in Figure 1. While this backbone is obviously incomplete in several ways, it is a realistic basis and it can be easily augmented. For example, routine processes like independent medical examinations or common law cases are not included but they easily can be — that is, insofar as they can be modelled at all. This also enables modularity, for example, a policy experiment might want to investigate different versions of the segmentation process — which is what we did in the prototype testing (see Section 3) — or alternative service decision deliberations.

2.3 Client Representation

The `Client` type, as mentioned, consists of two aspects (1) a history of states, and (2) an administrative event log which we identify as the client's claim. Consequently, a client object carries, at any point in the simulation, its entire history with it.

The history of states records the client's physical and mental health scores over time. In modelling terms, each state is just a vector of floating point variables. How many variables one wants to consider and to which metrics they correspond would depend on the scheme and the availability of data. The history of states is thus just a list of such vectors paired with the date at which that state is current.

The administrative event log is represented as an object of the `Claim` type. This object contains events, which are objects of type `Event`. Different sorts of events may happen to a client. Currently, we identify assessments, segmentation and service request events. An event has a date and an object detailing the change. For example, an assessment (a numerical score of a client's health state) event is simply a date with a score. A segmentation event also carries the tier and description of the service level the client is assigned to. A service request event carries a label describing the service requested, and the associated cost and workload.

Both the history of states and the claim objects are in essence just collections of date-state or date-change pairs. The framework provides a range of convenience functions to, for example, get a client's current state, segment, list of requests or the date of most recent change. Likewise there are convenience functions to add events. The user does not have to think about the internals of the `Client` object.

2.4 Conductor: Keeping Time and Score

To run simulations, GENSIMO, provides the `Conductor` type. A `Conductor` object contains the (1) initial and final dates of the simulation timeline, (2) a `Context` object for settings, parameter values and probability distributions and, (3) the cohort of `Client`'s. Specific implementations of GENSIMO then provide a `simulate!(conductor::Conductor)` function accepting a conductor to run the simulation on. As the simulation runs, the `Client` objects contained in the

`Conductor` are updated in-place, growing their histories of state and claims. When `simulation!()` returns, the `Conductor` object can be used to extract results. GENSIMO provides a range of convenience functions to extract information, for individual clients as well as scheme-level aggregate statistics. Various plotting tools are available that accept a `Conductor` or `Client` object directly.

The framework uses a daily time step and dates are represented using the `Date` type from the Julia package `Dates`, that is, not as naked integers. The `simulation!()` function thus steps through every day from the initial to the final date set in the `Conductor` object. Every client, however, is on a personal clock, as it were. For each client, the timing, number and kind of events depend on the characteristics of that client only, most prominently the day of coming on scheme. The detailed modelling of what event is to occur when to a client clearly depends on the scheme under consideration. We will therefore discuss the details of the event timing, especially the service request cycle, as part of the setting up of GENSIMO for the scheme the Transport Accident Commission in Victoria, Australia (Section 3.1).

3 Prototype — Setting up and Testing gensimo

We have been emphasising the design principle that GENSIMO be a *generic* framework. To test the framework, however, one does need to implement a particular scheme. This could be a hypothetical, stylised scheme of course, and there would actually be virtue in doing it that way but we find ourselves in the fortunate situation that an actual social insurance organisation, the Transport Accident Commission (TAC) provides part of the funding for the model development. The TAC also provide their ample expertise and experience as well as their data. Importantly, they also have research questions they would like a modelling perspective on.

Thus, in addition to our ambition to provide with GENSIMO a generic framework, we are concurrently developing a 'digital twin' for the TAC based on it. Our interpretation of 'digital twin' is a computer model that is an intuitive representation of the TAC insurance scheme(s), that is, it needs to be recognisable and realistic to the TAC themselves. Moreover, we want this model to be quantitatively realistic and accurate also — i.e. as 'close to the data' as possible. This means that we use the same state variables for our virtual clients' health states as the TAC uses internally, that they request the same (virtual) services as real clients at the same costs etc. To be clear, we do *not* use individual client data. What we use are aggregates, averages, distributions etc.

3.1 Setting up GENSIMO as a TAC Digital Twin

To describe the setting up of GENSIMO as a TAC Digital Twin, we will follow the backbone flow chart of Figure 1. In passing, we will elaborate how the various data structures, like `Client`, are operationalised.

Induction TAC clients have had a traffic accident. The accident date ('day zero') and the health state at that moment are the first entry in the `Client` object's history of states. As mentioned, the health state of a GENSIMO `Client` is just a vector of floating point variables. For the TAC Digital Twin, we choose this vector to correspond to their 'Big 6 complexities' which are claim-outcome predictor variables obtained from an internal longitudinal study. These Big 6 are complemented by a further 'Little 6'. Of these 12 variable, there are currently 3 that play a key role in the model:

- *Physical health* (φ) as a percentage of pre-accident health.
- *Mental health* (ψ) as a percentage of pre-accident health.
- *Satisfaction* with the scheme experience, as a percentage.

When creating a cohort of clients for a simulation, one could in principle populate these variables using the empirical probability distribution if data is available. For testing purposes we will be using uniformly random sampled clients. The model makes up a name, sex and age for a touch of realism, see Figure 2.

Induction is very much a silent process in the model. Clients have a day zero and corresponding health status at the moment of instantiation of their `Client` objects this essentially is all there is to it.

```
julia> client = Client()
Client ID: 5
  | Rebecca Matthews. 28 year-old female.
Status (2020-01-01):
  | ϕ = 0.7795488937126166
  | ψ = 0.2240473038573717
Claim (showing 0/0 newest events):
  | Empty claim.
```

Fig. 2. A random synthetic client, showing status (most recent health state) and claim events (none yet).

Segmentation The segmentation process, as currently modelled, consists of two events. The first is an assessment of the clients needs, based on health status. The second is the actual assignment of the client to an organisational division, with the concommitant level of care.

Assessment

Assessing the clients needs leads to a so-called Needs ID Score (NIDS), which is an integer between zero and 6, inclusive, with a higher number indicating greater needs. The model provides two ways of assigning a NIDS to a client, (1) based on their Big 6 values, which is following the TAC process closely, and (2) randomly based on the empirical distribution, which may be more realistic as currently only the first Big 2 are actively used in the model. NIDS screening is done in the model on the client's day zero, that is, when the simulated date reaches the client's day zero.

Assignment

Segmentation means that a client is assigned an appropriate service level ('Tier'), which corresponds to an organisational sub-division. Depending on a

client's segmentation, their case may be managed by as part of a single case manager's *portfolio* or as part of a *pool* of client assistance staff. Whether a client is managed in a pool or a portfolio, has consequences for the decision times for service requests and thus for client satisfaction and workload. Therefore, the segmentation strategy is a very important consideration in scheme design. As part of the prototype testing of the TAC Digital Twin, we have implemented two strategies:

BAU The current strategy. NIDS zero to 3, inclusive are segmented to Tier 1 while NIDS 4 and above go to Tier 2. By and large, Tier 1 are managed in pools Tier 2 in portfolios.

NEW Here, an intermediate tier is added, yielding NIDS 0, 1 and 2 going to Tier 1, NIDS 3 and 4 to Tier 2 and everything above going to Tier 3. By and large only Tier 1 is managed in pools.

Thus we expect the number of clients managed in portfolios to increase. The modelling question is whether this leads to an overall increase of workload, and if so, by how much.

It should be noted that this is a rather gross simplification of the actual segmentation process the TAC have. The actual process is not anything nearly so algorithmic and, moreover, we ignore entire parts of the client cohort here (notably clients with very severe injuries who are managed by a separate division altogether).

Service Request Cycle The bulk of a client's claim normally consists of service requests. When, how many and which services will be requested is here a key concern. As GENSIMO does not model a client's medical dynamics, the predicted claim evolution will be inaccurate at the individual level. Nonetheless, at the cohort level, clear patterns can be discerned in the data. Unsurprisingly, the closer to the accident, the more frequently a client can be expected to request a service. The intuitive explanation is simply that closer to the accident, the client *needs* those services as part of the recovery process. Ergo, the rate of service requests is inversely proportional to the health status of the client.

Empirically, we see clients request volume declining roughly exponentially over time and consequently, we assume *typical* recovery to be exponential as well, gradually approaching pre-accident health status. (See below, under 'Recovery'). Combining this with the request rate being proportional to the client's health, we obtain an inhomogeneous Poisson-like process. The intensity of the process being (1) inversely proportional to the instantaneous health state and, because of the client's recovery trajectory, (2) exponentially declining over time.

Service request

— Every day, from the date the client comes on scheme, a sample is drawn from the Poisson-like process described above. This sample is the number of requests placed on that day. If the number is zero, which it often is, the service request cycle is aborted and neither the claim nor the history of state

is updated (see below, under 'Recovery'). If the number is positive there are requests and these requests are drawn from the empirical distribution corresponding to the pool or portfolio environment the client is in. Thus, the kind of request (what service it is) as well as the cost and associated workload are in this sense random. Thus the client's claim will not make much medical sense. But at the cohort level the statistics ought to be realistic.

Service decision

— For each request, the scheme makes a service decision. The model makes the simplification that services are only ever accepted or denied. Moreover, this decision is logged immediately with the service request even though the associated workload may well exceed a single day. The workload is logged as part of the request so it still gets counted in the statistics as it should. The modelled version of the decision process is decidedly unsophisticated. Within the first 90 days after the accident, all requests are approved. After that, there is a flat rate of rejection of about 30%. While the 90 days blanket acceptance is not unrealistic, the flat rejection rate is and it should be considered a place holder, it is currently not informed by TAC data.

Iatrogenic effects

— The experience of clients with the scheme may have an impact on their recovery. At this point, we have included some hypotheses on these iatrogenic effects of the scheme. These are not validated with TAC data nor can we, at this point, adduce any e.g. psychological literature to substantiate. Again, they should be considered place holders. That said, we include a +1% physical health improvement for each approved request and a -1% psychological health deterioration as well as satisfaction for each denial.

Recovery As discussed above, clients' recoveries are modelled as exponential trajectories. The recovery *rate* is inversely proportional to the age of the client at date of the accident. This of course means recovery will be slower the older the client is.

It should be noted that the history of states of a `Client` object is updated only on dates when service requests are also made. Any iatrogenic effects (see above) are then applied to this updated health state.

Inactivation In principle, a TAC claim is never closed. If new services related to an accident are required, even if decades later, they become part of the client's existing claim. Such claims are relatively rare and most clients recover quickly (within months) and service requests cease, in which case a claim is considered 'inactive' 90 days after the last request. Thus, the number of *active* clients is an important monitoring instrument and the model provides functions to extract it in various ways: at a certain time, as a time series and by tier.

3.2 Testing and Comparison of Two Segmentation Strategies

As an initial, proof-of-concept type, analysis we set the model up to investigate the consequences of a new segmentation strategy (NEW) vis-à-vis the existing

one (BAU). We discussed the differences between these segmentation strategies in Section 3.1. The main modelling question is how the changed strategy will affect the insurers workload — both workload overall and the expected shifts of load from one organisational division to another.

Segmentation strategy, in the model, is related to workload in the following way:

- Segmentation leads to clients being assigned to service level tiers, based on their health status.
- Service level tiers (2 for BAU and 3 for NEW) correspond to a client being managed as by a *pool* of client assistance staff or in a single case manager's *portfolio*.
- The time between a service being requested and the scheme's decision on it is different depending on whether the client is in a pool or a portfolio.
- This time to decision is a proxy for the workload associated with the service request.

There is, empirically, no obvious pattern relating pool versus portfolio to more or less workload. That is, on an average neither pool nor portfolio is obviously faster to decide on service requests. No clear pattern was discernible when taking into account the nature of the service or when looking at how long after the accident a request was made. This suggests that by and large the *overall* workload ought to not change dramatically.

However, under the NEW strategy, a considerable class of clients that would have been assigned to a pool environment under the BAU strategy, are now assigned to portfolios. From data, we would expect the fraction of clients in portfolios to increase from about 24% to 44%. Thus we expect the workload to shift from pools to portfolios accordingly and the question is by how much.

Simulation Strategy We ran batch simulations with cohorts of 1,000 clients. Clients have uniformly random initial health parameters and age between 0 and 100 years. In other words, we are not attempting yet to model an actual cohort — just a proof of concept with few assumptions on the client parameters. We simulate the period from 1 January 2019 to 1 January 2029, inclusive. Clients come on scheme uniformly at random throughout that window, mimicking a constant flow of new clients into the scheme. Obviously, we are ignoring seasonal bursts, trends etc. We run 20 such simulations per scheme to obtain decent averages.

Summarising the main results, we find:

- Overall workload *decreases* by about 6%. A relatively small decrease (as anticipated) but not negligible — practically nor statistically — either.
- Workload in portfolios increases. With respect to the BAU the NEW strategy sees 74% more workload in porfolios.
- Workload in pools decreases. With respect to the BAU the NEW strategy sees 27% less workload in pools.

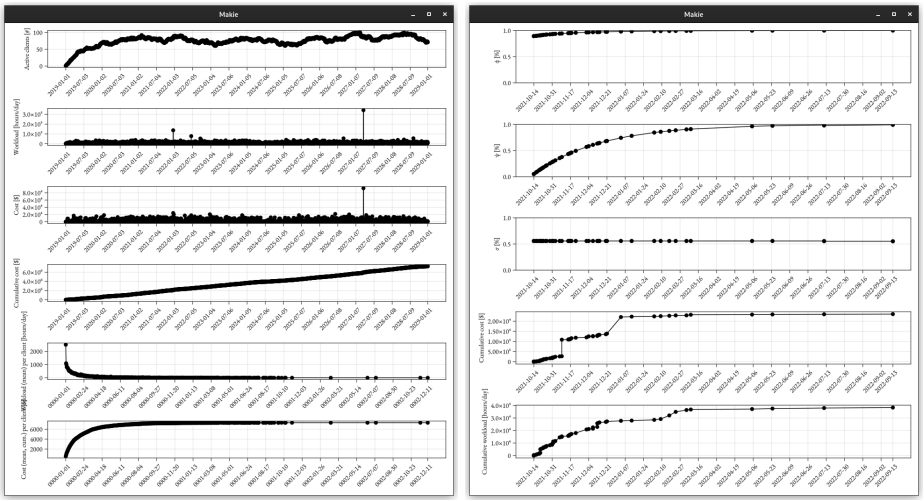

Fig. 3. Panel plots for simulations of a typical cohort (left) and a typical client (right).

– The balance between workload in pool versus portfolio shifts from 80/20 to 62/37 going from BAU to NEW.

We note that an individual simulation of 1,000 clients over 10 years (i.e. roughly 3,560 time steps) takes approximately 20 seconds on a laptop (Lenovo X1 Carbon running Fedora Linux 37 with a 12th Gen Intel Core i7-1255U @ 12x 4.7GHz CPU and 16GB RAM). In tests, we changed how clients entered the scheme over time from uniformly over the entire time frame to having them enter in a single burst in the first year. This changes the runtime to approximately 40 seconds. In the simulations for this paper, a single 20 simulation batch run thus takes in the order of 6–7 minutes.

4 Conclusions

This paper presented GENSIMO, a generic framework for modelling social insurance systems. We showed how it works by setting it up for a specific social insurance context, as a Digital Twin of the Transport Accident Commission (TAC) in Victoria, Australia. We then discussed early prototype testing which compared two segmentation strategies as to their impact on workloads for the insurer. Simulations are fast and at the current scale do not at all require any high-performance computing infrastructure. Initial results appear sensible though we do not claim this to constitute model verification.

When appraising the work presented, we should again stress that the framework, as well as the application as a digital twin, is under active development. Changes to the very structure of the generic framework are expected, as are

many changes and improvements to the digital twin. Likewise, the model has not gone through thorough verification and validation exercises. Model parameters, however, have typically been chosen in realistic ranges as suggested by the distributions gleaned from the data. In some areas there is scope for decent statistical matching of the empirical data and model parameters. In particular, the stochastic process governing the timing and volume of service request could be so calibrated.

Acknowledgements. The authors would like to gratefully acknowledge funding for this research from the NHMRC Centre of Research Excellence in Better Health Outcomes for Compensable Injury `https://cre-rfrti.centre.uq.edu.au/` and the Transport Accident Commission (TAC) `https://www.tac.vic.gov.au/`. Special thanks to our liaisons at TAC for their time, effort and input.

Disclosure of Interests. The authors have no other interests to disclose than those mentioned in the acknowledgements.

References

1. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing. SIAM review **59**(1), 65–98 (2017)
2. Datseris, G., Vahdati, A.R., DuBois, T.C.: Agents.jl: A performant and feature-full agent-based modeling software of minimal code complexity. Simulation p. 00375497211068820 (Jan 2022). https://doi.org/10.1177/00375497211068820
3. Thompson, J., Cruz-Gambardella, C.: Development of a Computational Policy Model for Comparing the Effect of Compensation Scheme Policies on Recovery After Workplace Injury. Journal of Occupational Rehabilitation **32**(2), 241–251 (Jun 2022). https://doi.org/10.1007/s10926-022-10035-w
4. Thompson, J., McClure, R., de Silva, A.: A Complex Systems Approach for Understanding the Effect of Policy and Management Interventions on Health System Performance, pp. 809–831 (Apr 2019). https://doi.org/10.1002/9781119485001.ch35
5. Thompson, J., McClure, R., Scott, N., Hellard, M., Abeysuriya, R., Vidanaarachchi, R., Thwaites, J., Lazarus, J.V., Lavis, J., Michie, S., Bullen, C., Prokopenko, M., Chang, S.L., Cliff, O.M., Zachreson, C., Blakely, A., Wilson, T., Ouakrim, D.A., Sundararajan, V.: A framework for considering the utility of models when facing tough decisions in public health: A guideline for policy-makers. Health Research Policy and Systems **20**(1), 107 (Oct 2022). https://doi.org/10.1186/s12961-022-00902-6